# Better Approximation of Betweenness Centrality

Student thesis

at

Institute for Theoretical Computer Science, Algorithmics II
Universität Karlsruhe (TH)

of

Robert Geisberger

supervised by:

Prof. Dr. Peter Sanders
Dominik Schultes

## Abstract

Centrality indices are used to classify a graph in important and unimportant vertices. A commonly used index is *betweenness centrality* which is based on shortest paths. For large graphs it is almost impossible to calculate exact betweenness centrality in appropriate time. The best known algorithm today is based on solving $n$ SSSPs and requires $O(nm + n^2 \log(n))$ time. But in most cases, e.g. on a home computer, resources are limited. Current approximation algorithms extrapolate values by solving only $k \ll n$ SSSPs. Vertices near the source of a SSSP are often overestimated. We introduce improvements which take the distance to the source into account. Euclidean distance between approximation and exact values is reduced by factor 4 with same runtime. Or runtime is 16 times faster with same Euclidean distance in a standard example, the movie actor network. Other real-world networks show similar promising results.

# Contents

# 1 Introduction

## 1.1 Motivation

Real-world networks have been a field of study and research for a long time. They are represented by a graph $G = (V, E)$ of vertices $V$ and edges $E$. Often they have additional properties, e.g. weighted edges, undirected edges or parallel edges. A common example is the internet router topology where routers are vertices and links between routers are edges. One would like to know which routers or which links are *important*, e.g. how *severe* is the breakdown of a specified router or link. So *centrality measures* are required to label each vertex or edge with a number indicating its importance. But there is neither a mathematical definition for *important* nor for *severe*. So since the 1950's many centrality indices have evolved, each with specific applications. Some examples for applications include the facility location problem, highway-node routing, web page ranking or prediction of polls. A centrality index is a structural index for vertices or edges. Often they are based on shortest paths. Some examples are closeness centrality, stress centrality, graph centrality, reach centrality and betweenness centrality [4]. Our attention belongs to the general betweenness centrality and a derivate that we introduce in this paper, canonical betweenness centrality. It can be applied to vertices or edges. Both indices measure to what extend a vertex or edge is 'in between groups of vertices'.

## 1.2 Definition

Let $G = (V, E)$ be a *graph*, where $V$ is a set of *vertices* and $E$ is a multiset of *edges*. Let $\omega : E \to \mathbb{R}^{>0}$ be a *weight function* for this graph with positive weights. Let $d_G(s, t)$ be the shortest path distance between $s, t \in V$. Denote $SP_{st}$ the set of different shortest path between $s, t \in V$ and $\sigma_{st} := |SP_{st}|$. For $v \in V$ denote $SP_{st}(v)$ the set of different shortest path containing $v$ with $s \neq v \neq t$, and $\sigma_{st}(v) := |SP_{st}(v)|$.

### General Betweenness Centrality

General betweenness centrality [9], [1] for a vertex $v \in V$ adds up fractions of shortest paths.

$$c_B(v) := \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

### Canonical Betweenness Centrality

Canonical betweenness centrality is a derivate of betweenness centrality. It considers only one shortest path between every $s, t \in V$. Define a set $\mathcal{P}$ of *canonical paths*, that is a set $\mathcal{P}$ that meets the following three conditions:

$$\mathcal{P} \subseteq \bigcup_{s,t \in V} SP_{st} \tag{2}$$

$$\forall s, t \in V : |\mathcal{P} \cap SP_{st}| \in \{0, 1\} \tag{3}$$

$$\forall s, t \in V : (|\mathcal{P} \cap SP_{st}| = 0 \Leftrightarrow |SP_{st}| = \sigma_{st} = 0) \tag{4}$$

For $s, t, v \in V$ denote

$$\sigma^*_{st}(v) := \begin{cases} 0 & \text{if } \mathcal{P} \cap SP_{st}(v) = \emptyset \\ 1 & \text{if } \mathcal{P} \cap SP_{st}(v) \neq \emptyset \end{cases} \tag{5}$$

$$c_C(v) := \sum_{s,t \in V} \sigma^*_{st}(v) \tag{6}$$

The idea behind canonical betweenness centrality is routing. Even if there is more than a single path between two endpoints $s$ and $t$, the routing software usually finds only one. Some advanced routing algorithms like highway-node routing [16] calculate shortest-path overlay graphs in advance to speed up the actual routing algorithm. These overlay graphs consist of a subset of vertices of $G$ and only those edges that are necessary for shortest paths between them. As soon as a search for a route between $s$ and $t$ reaches a vertex in the overlay graph only vertices and edges in the overlay graph are considered. Therefore the chosen subset of vertices is crucial to time requirements of the search. Canonical betweenness centrality can be used to select this subset.

### Other important indices

Some other important indices based on shortest paths are listed below, but those indices are currently of no interest for us.

| | |
|---|---|
| $c_O(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$ | Closeness centrality [15] |
| $c_G(v) = \frac{1}{max_{t \in V} d_G(v,t)}$ | Graph centrality [11] |
| $c_S(v) = \sum\limits_{s \neq v \neq t \in V} \sigma_{st}(v)$ | Stress centrality [17] |
| $c_R(v) = \max_{s,t \in V, \sigma_{st}(v)>0} \min(d_G(s,v), d_G(v,t))$ | Reach centrality [10] |

## 1.3 Related work

The fastest algorithm up to date to calculate exact general betweenness centrality is Brandes' exact algorithm [3]. It requires $O(nm+n^2\log(n))$ time for a weighted graph where $n = |V|, m = |E|$. It solves $n$ SSSP (Single Source Shortest Path) problems with Dijkstra's algorithm. Then it adds counter values from leaves to the top. For large graphs, e.g. a street graph of Western Europe with approximately 18 million vertices and 22 million edges, exact calculation is almost unfeasible with only a small amount of time and computing power. Brandes and Pich [5] introduced a unbiased technique to approximate general betweenness by choosing only $k \ll n$ pivots as source for the SSSP algorithm. They turned their attention especially to the pivot selection method. As their results show no significant advantage of specialized methods over random pivot selection no other selection method was used by us. Another advantage of random selection is its performance.
Bader and Madduri [2] introduced parallel algorithms to calculate exact general betweenness centrality. Their algorithms can reduce execution time by some multipliers but need many CPU cores and much RAM. This is useful if exact general betweenness centrality values are necessary and resources are available. But if good estimations are sufficient or resources are too expensive, approximation algorithms should be preferred.

## 1.4 Our contribution

Brandes' method randomly chooses $k \ll n$ vertices as sources to solve the SSSP. These vertices are called *pivots*. The centrality of vertices near those pivots is often overestimated. To alleviate this problem we use the distance between a vertex and the pivot in our approximation methods. We introduce two different approaches to take the distance into account. Test results for different graph types and sizes show advantages of our new algorithms, they always perform better within the same time frame. We will prove that our estimators are unbiased and have comparable analytical error bounds as Brandes' method. Then we map our algorithms to general betweenness centrality and show similar results.

## 1.5 Outline

First, approximation of canonical betweenness centrality is described in Section 2. We present our new methods (2.1), show how to implement them efficiently (2.2), introduce our test instances and compare it with Brandes' method (2.3). In Section 3 our methods are adapted to general betweenness centrality. A slightly new definition of the methods (3.1) and implementation of algorithms (3.2) is necessary. We use another test instance but show similar results regarding Brandes' method (3.3).

# 2 Approximation of Canonical Betweenness Centrality

First we will create an abstract approximation framework. All approximation methods within this framework will be unbiased. We show how to integrate Brandes' method in this framework and introduce our new variants, bisection method and linear scaling. After that, efficient implementations of all three methods are proposed and experimental results are shown.

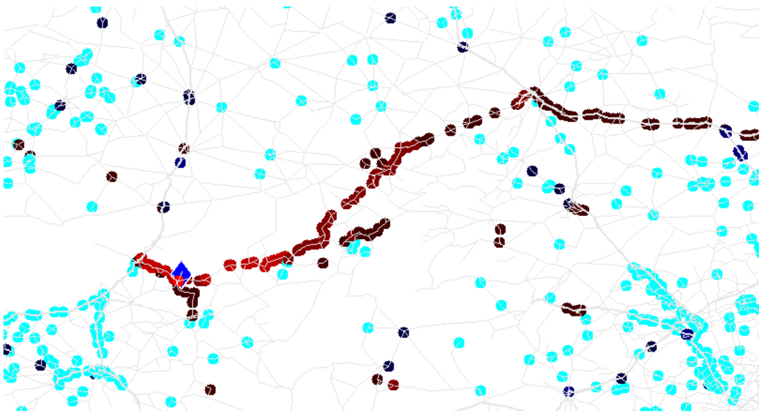


Figure 1: A part of the road network of Spain, 16 pivots compared to 8192 pivots with the bisection method, the blue diamond is a backward pivot, cyan dots are underestimated by more than factor $10^{-8}$, red dots are overestimated, the darker the red the lower is the overestimation.

## 2.1 Methods

Brandes does not take distance between pivot $s$ and vertex $v$ into account, as illustrated in Figure 1. Vertices near a pivot are often overestimated.

Our framework will depend on a distance function $d : V \times V \to \mathbb{R}^{\geq 0}$ satisfying

$$\forall v, s, t \in V : (\sigma^*_{st}(v) = 1 \Rightarrow d(s,t) = d(s,v) + d(v,t)) \tag{7}$$

For example the shortest path distance $d_G$ is valid. Let $f_\ell : [0,\ell] \to \mathbb{R}$ be a function to weight the contribution of a forward pivot according to the distance $\ell$. To get an unbiased estimator, the contribution of a backward pivot is weighted with

$$\bar{f}_\ell := x \mapsto 1 - f_\ell(x). \tag{8}$$

This introduces the possibility to count contributions depending on the distance between $v$ and $s$ or $t$. Following Brandes' approach in [3], we define for $s, t, v \in V$ the contribution $\delta^*_{s\bullet}(v)$ of a *forward pivot* $s \in V$ to the canonical betweenness centrality value of $v$ is and the contribution $\delta^*_{\bullet t}(v)$ of a *backward pivot* $t \in V$.

$$\delta^*_{s\bullet}(v) := \sum_{t \in V} \left( \sigma^*_{st}(v) \cdot f_{d(s,t)}(d(s,v)) \right) \tag{9}$$

$$\delta^*_{\bullet t}(v) := \sum_{s \in V} \left( \sigma^*_{st}(v) \cdot \bar{f}_{d(s,t)}(d(v,t)) \right) \tag{10}$$

To extrapolate an unbiased estimator, we define a random variable $X$ with uniform distribution among all $2n$ events ($n$ forward pivots and $n$ backward).

$$X = \begin{cases} 2n \cdot \delta^*_{s\bullet}(v) & \text{if forward pivot } s \text{ is chosen} \\ 2n \cdot \delta^*_{\bullet t}(v) & \text{if backward pivot } t \text{ is chosen} \end{cases} \tag{11}$$

**Lemma 1** *$X$ is an unbiased estimator meaning $\forall v \in V : E(X) = c_C(v)$ for all methods constructed satisfying conditions listed above.*

*Proof.*

$$\begin{aligned} E(X) &\overset{(11)}{=} \frac{1}{2n} \cdot \left( \sum_{s \in V} (2n \cdot \delta^*_{s\bullet}(v)) + \sum_{t \in V} (2n \cdot \delta^*_{\bullet t}(v)) \right) \\ &\overset{(9),(10)}{=} \sum_{s \in V} \sum_{t \in V} \left( \sigma^*_{st}(v) \cdot f_{d(st)}(d(s,v)) \right) + \sum_{t \in V} \sum_{s \in V} \left( \sigma^*_{st}(v) \cdot \bar{f}_{d(s,t)}(d(v,t)) \right) \\ &= \sum_{s,t \in V} \sigma^*_{st}(v) \left( f_{d(s,t)}(d(s,v)) + \bar{f}_{d(s,t)}(d(v,t)) \right) \\ &\overset{(7),(8)}{=} \sum_{s,t \in V}^{n} \sigma^*_{st}(v) \\ &\overset{(6)}{=} c_C(v) \end{aligned}$$

□

## Brandes' Method

Brandes' method does not depend on the distance between $v$ and $s$ or $t$; so $f_\ell$ and $\bar{f}_\ell$ are constant. There is no necessity to define a $d$ function.

$$f_\ell := x \mapsto \frac{1}{2} \tag{12}$$

$$\bar{f}_\ell := x \mapsto \frac{1}{2} \tag{13}$$

We observe that Brandes did not use backward pivots. But to keep all methods comparable we chose to add them.

Now we introduce two different methods that take shortest path distance between $v$ and $s$ or $t$ into account to achieve better results.

## Linear Scaling

Often vertices near a selected pivot $p$ are overestimated as Figure 1 illustrates. Linear approximation will soften this phenomenon. The nearer $v$ to $s$ on forward search or to $t$ on backward search the lower a contribution counts.

$$f_\ell := x \mapsto \frac{x}{\ell} \tag{14}$$

$$\bar{f}_\ell = x \mapsto 1 - \frac{x}{\ell} \tag{15}$$

The $d$ function is the shortest path distance $d_G$ between two vertices. Conditions (7), (8) are satisfied.

## Bisection Method

The bisection method has the same goal as linear scaling but does not use a linear approach. Only contributions to $c_C(v)$ are taken into account if $v$ is at least half distance between $s$ and $t$ away from $s$ on forward search or more than half distance away from $t$ on backward search. The $d$ function is the unit distance. Usually unit distance between two vertices depends on the chosen path. We use the canonical shortest path, if existent, to measure unit distance. This approach satisfies conditions (7), (8).

$$f_\ell := x \mapsto \begin{cases} 1 & \text{if } x \geq \frac{\ell}{2} \\ 0 & \text{if } x < \frac{\ell}{2} \end{cases} \tag{16}$$

$$\bar{f}_\ell = x \mapsto \begin{cases} 1 & \text{if } x > \frac{\ell}{2} \\ 0 & \text{if } x \leq \frac{\ell}{2} \end{cases} \tag{17}$$

In Figure 2 all three methods are compared by their definition of $f_\ell$.

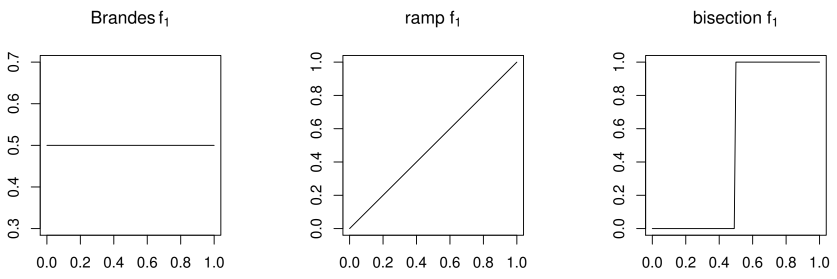


Figure 2: Comparison of function $f_1$

## Other methods

Within our framework other methods could be defined, too. For example the sigmoid function that is used in neural networks, could be a good idea. Or use bisection method with shortest path distance. But it is difficult or impossible to implement them efficiently, i.e. in O(SSSP).

## Error bounds

Error bounds of Brandes [5] can be adopted. Hoeffding [13] proves that for independent identically distributed random variables $X_1, \ldots, X_k$ with $0 \leq X_i \leq M$ and arbitrary $\xi \geq 0$,

$$prob\left(\left|\frac{X_1 + \ldots + X_k}{k} - E\left(\frac{X_1 + \ldots + X_k}{k}\right)\right| \geq \xi\right) \leq 2e^{-2k\left(\frac{\xi}{M}\right)^2} \tag{18}$$

Let $X_i$ represent contribution to $c_C(v)$ for pivot number $i$. Because pivots are chosen independently, random variables $X_1, \ldots, X_k$ are independent and uniformly distributed. Each vertex is on at most $(n-2)$ shortest paths because source and target of a shortest path do not count. Extrapolation yields factor $2n$. A factor $\alpha$ is needed to take the behavior of $f_\ell$ and $\bar{f}_\ell$ into account.

$$\alpha := \max_{x \in [0,\ell]} (f_\ell(x), \bar{f}_\ell(x))$$

Applying Hoeffdings bound with

$$M := 2n(n-2) \cdot \alpha \tag{19}$$

$$\xi := \epsilon \cdot 2(n-2) \cdot \alpha \tag{20}$$

yields an error bound from above by $\xi$ with probability $2e^{-2k\left(\frac{\epsilon}{n}\right)^2}$. For Brandes' method $\alpha = \frac{1}{2}$, for our new methods $\alpha = 1$. Our methods estimate contributions depending on the distance to the pivot. This leads to an higher upper bound $M$. Thus, with Hoeffdings inequality we can only prove a weaker error bound than for Brandes' method but we will show that our methods are better in practice.

## 2.2 Algorithms

Our new algorithms are based on Brandes' algorithm [5]. We will shortly introduce Brandes' algorithm and then focus on the changes made to implement the bisection method and linear scaling.

### Brandes' Algorithm

All algorithms have in common that they need to solve the SSSP problem resulting in a *shortest path tree* with root $s$ on forward search (or root $t$ on backward search). For unweighted graphs this is possible with breath first search (BFS), for weighted graphs Dijkstra's algorithm is used. For backward search the opposite graph $G^{op}$ is used, having the same vertices but all edges have switched source and target. To get an efficient implementation, Brandes proved a recursion equation to calculate contribution $\delta^*_{s\bullet}(v)$ out of contribution of its children in the shortest path tree. Let $C_s(v)$ be the multiset of *children* of $v \in V$ in the shortest path tree. Then,

$$\delta^*_{s\bullet}(v) = \sum_{w \in C_s(v)} (1 + \delta^*_{s\bullet}(w)) \tag{21}$$

If $w$ is a child of $v$, then $v$ lies on all shortest paths that $w$ lies on plus the one between $s$ and $w$. Same thing for other children $x$ of $v$ as Figure 3 illustrates.

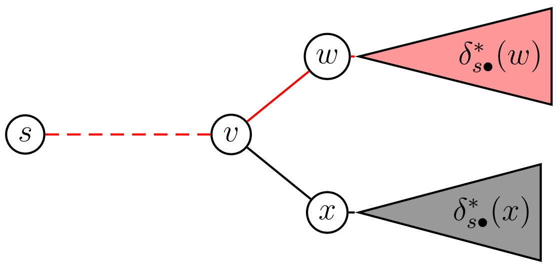


Figure 3: Visualization of Brandes' method

The recursion equation is also valid for backward search because the proof can be applied to the opposite graph $G^{op}$ as well.

In Algorithm 1 Line 11, the recursion equation is used. Let *cSSSP* denotes the CanonicalSSSP algorithm of Line 5. Algorithm Canonical-Brandes requires $O(cSSSP) + O(n)$ time per pivot. $O(cSSSP)$ is not the same as $O(SSSP)$, we will elaborate this later in this section.

## Linear Scaling Algorithm

Linear scaling only needs small modifications to Brandes' algorithm. Because it uses shortest path distance to weight contributions, this distance needs to be stored in the shortest path tree.

Why does Algorithm 2 implement linear scaling? We will show this for the forward direction, the backward direction is similar. Identify $c_{local}[v]$ with $\frac{\delta^*_{s\bullet}(v)}{d(s,v)}$. Brandes' recursion equation (21) needs to be modified to add up fractions $\frac{1}{d(s,t)}$ instead of 1.

$$\delta^*_{s\bullet}(v) \overset{(9),(14)}{=} \sum_{t\in V} \left(\sigma^*_{st}(v)\cdot \frac{d(s,v)}{d(s,t)}\right) = d(s,v)\cdot \sum_{t\in V} \frac{\sigma^*_{st}(v)}{d(s,t)}$$

$$\Rightarrow \sum_{w\in C_s(v)} \frac{1+\delta^*_{s\bullet}(w)}{d(s,w)} = \sum_{w\in C_s(v)} \left(\frac{1}{d(s,w)} + \sum_{t\in V}\frac{\sigma^*_{st}(w)}{d(s,t)}\right) \overset{(*)}{=} \sum_{t\in V}\frac{\sigma^*_{st}(v)}{d(s,t)} = \frac{\delta^*_{s\bullet}(v)}{d(s,v)}$$

$(*)$: same argument as illustrated in Figure 3.
So $c_{local}[v] = \sum_{w\in C_s(v)} \left(c_{local}[w] + \frac{1}{d(s,w)}\right)$ in Line 11 and $\delta^*_{s\bullet}(w) = c_{local}[w]\cdot d(s,w)$ in Line 9.

Canonical-Linear-Scaling requires $O(cSSSP) + O(n)$ time per pivot like Canonical-Brandes.

## Bisection Algorithm

The bisection algorithm processes the shortest path tree in a depth first search (DFS) manner so it can maintain a path from $s$ to $w$ on a stack $S$. Random access to this stack allows calculation of the middle in $O(1)$. For a DFS, children are needed in the shortest path tree to traverse the tree from the root to the leafs. Usually Dijkstra's algorithm only stores parents in the shortest path tree so a modification to Dijkstra's algorithm is necessary.

Except for Lines 9-14, Algorithm 3 implements Brandes' method. But instead of processing vertices in nondecreasing distance to $s$, the shortest path tree is traversed depth first and counters are incremented on the way back, when a *pop* occurs. The DFS is only used to adapt the means to the end. The 'bisection trick' happens in Line 14. Subtracting 1 from the counter in the middle will recursively neutralize +1 of Line 24 or 31. So $\forall t \in V : \sigma^*_{st}(v)$ will not contribute to $c_C(v)$ if $d(s,v) <= m$ satisfying the definition of the bisection method.

Canonical-Bisection requires $O(cSSSP)+O(n)$ time per pivot. The data structure to store the search tree is a vertex-array storing *first child* and *next sibling*, illustrated in Figure 4. Because each vertex $v \in V$ occurs in the shortest path tree at most once and either as first child or as next sibling, Lines 16 and 32 are executed at most $n$ times. So lines 21 and 28 will be executed at most $n$ times. The *while* loop starting at line 8 will be executed at most $n$ times because either there is a first child, another vertex has a next sibling or $S$ is empty and the loop terminates. The inner while loop starting at line 18 always executes a *pop*. So except for Line 5, the algorithm is in $O(n)$.

**Algorithm 1**: Canonical-Brandes

```
Input: Graph $G = (V, E)$ with $n = |V|$, number of pivots $k$
Output: Vertex-array $c$ of estimated canonical betweenness centrality values for each $v \in V$
foreach $v \in V$ do
    $c[v] \leftarrow 0$;
for $i = 1$ to $k$ do
    Choose $s \in V$ and search direction uniformly at random;
    Solve CanonicalSSSP($s$,$direction$);
    foreach $v \in V$ do
        $c_{local}[v] \leftarrow 0$;
    forall $w \in V$ reachable from $s$, $w \neq s$ in order of non-increasing distance to $s$ do
        $c[w] = c[w] + c_{local}[w]$;
        $v \leftarrow$ parent of $w$;
        $c_{local}[v] \leftarrow c_{local}[v] + c_{local}[w] + 1$ ; // applying recursion equation (21)
// extrapolate
foreach $v \in V$ do
    $c[v] \leftarrow c[v] \cdot \frac{2n}{k}$
return $c$
```

**Algorithm 2**: Canonical-Linear-Scaling

```
Input: Graph $G = (V, E)$ with $n = |V|$, number of pivots $k$
Output: Vertex-array $c$ of estimated canonical betweenness centrality values for each $v \in V$
foreach $v \in V$ do
    $c[v] \leftarrow 0$;
for $i = 1$ to $k$ do
    Choose $s \in V$ and search direction uniformly at random;
    Solve CanonicalSSSP($s$,$direction$);
    foreach $v \in V$ do
        $c_{local}[v] \leftarrow 0$;
    forall $w \in V$ reachable from $s$, $w \neq s$ in order of non-increasing distance do
        $c[w] = c[w] + c_{local}[w] \cdot d(s, w)$;
        $v \leftarrow$ parent of $w$;
        $c_{local}[v] \leftarrow c_{local}[v] + c_{local}[w] + \frac{1}{d(s,w)}$; // applying modified recursion equation
// extrapolate
foreach $v \in V$ do
    $c[v] \leftarrow c[v] \cdot \frac{2n}{k}$
return $c$
```

```
Algorithm 3: Canonical-Bisection
Input: Graph $G = (V, E)$ with $n = |V|$, number of pivots $k$
Output: Vertex-array $c$ of estimated canonical betweenness centrality values for each $v \in V$
1  foreach $v \in V$ do
2      $c[v] \leftarrow 0$
3  for $i$ = 1 to $k$ do
4      Choose $s \in V$ and search direction uniformly at random;
5      Solve CanonicalSSSPWithChildren($s$,$direction$);
6      $S \leftarrow \emptyset$;   // stack with random access, containing path to $s$, items ($v$,$c_{local}[v]$)
7      push($S$,$(s, 0)$);
8      while $S \neq \emptyset$ do
           // count for vertex in the middle only on forward search
9          if forward search then
               // $m$ is the index of the first vertex $v$ with $f_{d(s,\mathtt{back}(S))}(d(s, v)) = 0$
10             if size($S$)$= 1$ then $m \leftarrow 0$ else $m \leftarrow \left\lfloor \frac{\mathtt{size}(S)-2}{2} \right\rfloor$;
11         else// on backward search $\bar{f}_\ell$ is decisive
12             $m \leftarrow \left\lfloor \frac{\mathtt{size}(S)-1}{2} \right\rfloor$;
13         $(v, a) \leftarrow S[m]$;
14         $S[m] \leftarrow (v, a - 1)$; // ‘bisection trick’
15         if back($S$) has first child $v$ in the shortest path tree then
16             push($S$,$(v, 0)$);
17         else
               // remove vertices from stack with no next sibling and increment
                  counters
18             while $S \neq \emptyset$ and back($S$) has no next sibling $v$ in the shortest path tree do
19                 $(v, a) \leftarrow$back($S$);
20                 $c[v] \leftarrow c[v] + a$;
21                 pop($S$);
22                 if $S \neq \emptyset$ then
23                     $(w, b) \leftarrow$back($S$);
24                     back($S$)$\leftarrow (w, b + a + 1)$; // applying recursion equation (21)
               // Remove vertex with sibling as well, but push sibling on stack
25             if $S \neq \emptyset$ then
26                 $(v, a) \leftarrow$back($S$);
27                 $c[v] \leftarrow c[v] + a$;
28                 pop($S$);
29                 if $S \neq \emptyset$ then
30                     $(w, b) \leftarrow$back($S$);
31                     back($S$)$\leftarrow (w, b + a + 1)$; // applying recursion equation (21)
32                 push($S$,(next sibling of $v$, 0))
33 foreach $v \in V$ do
34     $c[v] \leftarrow c[v] \cdot \frac{2n}{k}$; // extrapolate
35 return $c$
```

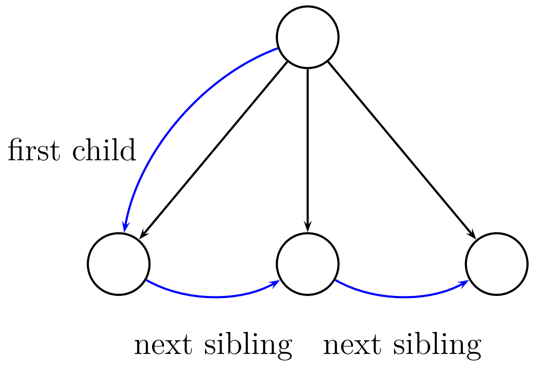


Figure 4: First child and next sibling in shortest path tree

Using shortest path distance instead of unit distance seems more suggesting. But this will likely not be in $O(n)$. We give an example in Figure 5; middle vertex for $b_i$ is $a_1$ and for $c_i$ is $a_r$. As the DFS search proceeds from $b_i$ to $c_i$ it needs to relocate the middle from $a_1$ to $a_r$. Doing this step by step from $a_1$ to $a_2$, ...to $a_r$ requires $\Omega(r)$ steps, thus the total execution time is in $\Omega(r^2) = \Omega(n^2)$. This is not a proof that the bisection method with shortest path distance

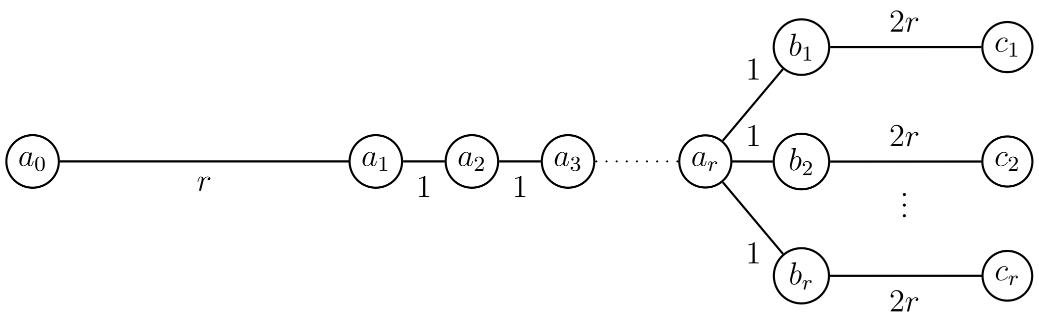


Figure 5: Example for bisection method with shortest path distance

is impossible in $O(n)$, but it shows that walking from one vertex to the next one to locate the middle vertex is not in $O(n)$.

**Canonical paths**

So far we implicitly assumed that a graph $G$ has unique shortest paths. To allow estimation of canonical betweenness centrality on any graph a selection method is needed to select unique shortest paths. This is done by forming a tree out of the acyclic shortest path graph with root $s \in V$. Each vertex is represented by an index. If there is a vertex in the shortest path graph that can have more than one parent on its shortest path to $s$, the parent with the lowest index value is chosen. This approach is efficient if there is only forward search.

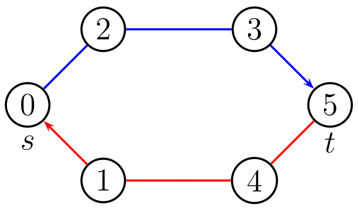


Figure 6: Example for canonical paths on backward search.

To find same unique shortest paths on backward search modifications are necessary. These are motivated by Figure 6; the presented graph has unit distance. Forward search starting at vertex $s$ will find the blue path. As Dijkstra search reaches vertex 5, two shortest path with distance 3 are possible, either $0-2-3-5$ or $0-1-4-5$. Because 3 is lower than 4 the blue path is chosen. Backward search will find the red path. To get correct results, both searches need to find the same path. Our solution is not efficient in theory, but works well in practice. We use the mentioned simple approach on forward search and locate the same path on backward search. Thus if backward search reaches vertex 0 and observes that there are two possible shortest path, either $5-3-2-0$ or $5-4-1-0$, it needs to traverse back to the next vertex both paths have in common to decide which path to choose; in this case it is vertex 5.

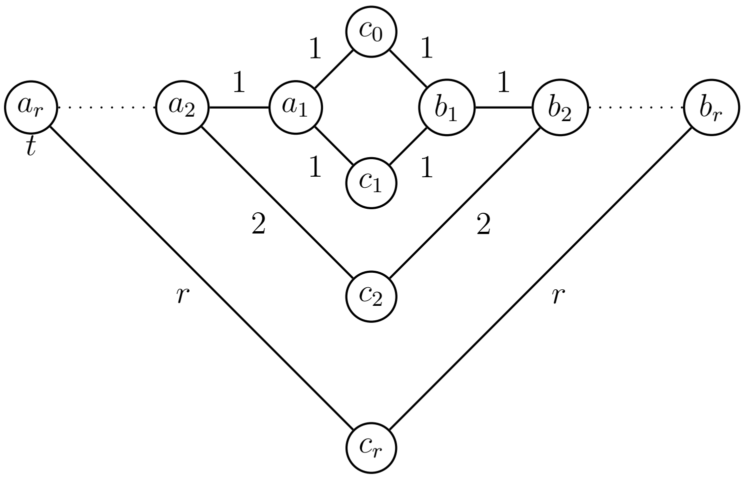


Figure 7: Second example for canonical paths on backward search.

Figure 7 is an example that this is not in $O(SSSP)$. This graph has $n = 3r + 1$ vertices, edges are labeled with their weight. Starting backward search at vertex $t$ the first vertex with more than one shortest path is $b_1$. It needs to traverse back four edges to reach vertex $a_1$. For $b_2$ six edges need to be traversed back. For $b_r$, $2 + 2r$ edges need to traversed back. In total this adds up to $2r + r(r + 1) \in \Omega(n^2)$ edges. So canonical betweenness centrality should only be used if there are only few shortest paths between two vertices and if two shortest paths of same source and target differ only on short parts. Street graphs are a good example for such graphs and our tests show insignificant differences between duration of Dijkstra's algorithm and our modified version. For example, the street graph of Belgium, our main test instance, has for each pair at most 16 different shortest paths but most paths are canonical.

## 2.3 Experiments

### Environment and Instances

The experiments were done on one core of a single AMD Opteron Processor 270 clocked at 2.0 GHz with 8 GB main memory and 2 × 1 MB L2 cache, running SuSE Linux 10.0 (kernel 2.6.13). The program was compiled by the GNU C++ compiler 4.0.2 using optimization level 3.

All test results for canonical betweenness centrality are calculated with a directed road network of Belgium with $n = 463\,514$ and $m = 596\,119$. Some visualizations originate from a road

| algorithm | seconds/pivot |
|---|---|
| Brandes | 0.502 |
| bisection (unit) | 0.539 |
| bisection (shortest path) | 0.561 |
| linear scaling | 0.508 |

Table 1: Runtime of different algorithms that approximate canonical betweenness centrality

network of Western Europe with $n = 18\,029\,721$ and $m = 22\,413\,128$. Both networks have been made available for scientific use by the company PTV AG. The original graphs contain for each edge a length and a road category. We assign average speed to each road category, compute average travel times, and use them as weight.

The algorithms described in Section 2.2 were implemented using custom data structures for graphs, same as in [16]. Basically an adiacency array is used.

*Number of pivots* refers to number of pivots that Brandes' algorithm used. All other algorithms get at most as much time as Brandes' algorithm to calculate their results. So faster algorithms like Brandes' are not handicapped by more sophisticated but slower algorithms. We want to test our algorithms by using all $n$ vertices as pivots. This is a good test for correctness. So we do not randomly choose search direction and do not put back pivots after they were chosen, i.e. a chosen pivot cannot be chosen again. Also the number of pivots specifies the number of forward and of backward pivots, e.g. 1024 pivots indicate 1024 forward pivots and 1024 backward pivots. So with $n$ pivots the exact canonical betweenness centrality is calculated because every vertex was once forward pivot and once backward pivot and all estimators are unbiased.

We start our analysis by comparing runtimes of the algorithms in Table 1. They are quite similar, but the bisection algorithm, especially with shortest path distance, is a little bit slower.

In Figures 8 and 9 methods are compared for different number of pivots. The first two global scores directly compare estimated values to exact values. The Euclidean distance is between $c_C^{approx}$ and $c_C^{exact}$ viewed as normalized vectors in $\mathrm{I\!R}^n$, same as [5].

$$\sqrt{\sum_{v\in V}\left(\frac{c_C^{approx}(v)}{\sum\limits_{v\in V} c_C^{approx}(v)} - \frac{c_C^{exact}(v)}{\sum\limits_{v\in V} c_C^{exact}(v)}\right)^2}$$

The geometric mean of relative error takes only vertices $v$ with $c_C^{approx}(v) \neq 0$ into account, to avoid division by zero. The number of vertices $v$ with $c_C^{approx}(v) = 0$ is given separately.

$$V' := \{v \in V \mid c_C^{approx}(v) \neq 0\}$$

$$\sqrt[|V'|]{\prod_{v\in V'} \max\left(\frac{c_C^{approx}(v)}{c_C^{exact}(v)}, \frac{c_C^{exact}(v)}{c_C^{approx}(v)}\right)}$$

Both global scores measure the quality of the approximations. But compared to the Euclidean distance, the geometric mean of relative error stronger regards the variance of the values.

The Euclidean distance and the geometric mean of relative error show good results for all methods. Lower values are better, meaning smaller errors. Errors decrease with growing number of pivots. The bisection method and linear scaling are constantly better compared to Brandes' method. For example the bisection method compared to Brandes' method improves the Euclidean distance by factor $1.7 - 2.7$. Or the bisection algorithm is $\approx 6$ times faster than Brandes' algorithm.

Centrality indices are often used to classify vertices or to establish an order among them. We assigned a rank $r_C(v)$ between 1 and $n$ to each vertex $v$, the higher the rank the smaller the canonical betweenness centrality value. The other two global scores compare the estimated rank to the exact rank. The inversion number counts the number of pairs of vertices that are in wrong order.

$$\left|\left\{\{v,w\} \mid r_C^{exact}(v) > r_C^{exact}(w) \text{ and } r_C^{approx}(v) < r_C^{approx}(w)\right\}\right|$$

The geometric mean of relative rank error shows similar behavior as the inversion number.

$$\sqrt[n]{\prod_{v\in V} \max\left(\frac{r_C^{approx}(v)}{r_C^{exact}(v)}, \frac{r_C^{exact}(v)}{r_C^{approx}(v)}\right)}$$

Again, our new methods show better results than Brandes' method. The score values of Brandes' method increase with increasing number of pivots. But because ranks are compared instead of estimated values, increasing errors are no error in the implementation of our algorithms. With increasing number of pivots both scores eventually decrease as Figure 10 proves. The local maximum in both scores is rather a property of the used road network. In Figure 11 the inversion number and the geometric mean of relative rank error are shown for a random graph without the anomalies observed for the Belgian road network. The graph is generated like the random graph Brandes [5] used. It is unweighted, undirected, has 1 000 vertices and 10 074 edges. Each of the $\binom{n}{2}$ edges was chosen independently with probability $\frac{10\,000}{\binom{n}{2}}$. The bisection method and linear scaling show strictly decreasing inversion number and geometric mean of relative rank error with the bisection method ahead.

In Figure 12 vertices are classified in levels to get a better overview. Level 11 contains the vertices with rank 1-128, level 10 the next 256 vertices having rank 129-384, descending in an logarithmic manner. The *relative error*

$$\frac{c_C^{approx}(v)}{c_C^{exact}(v)}$$

suggests that bisection method is even better than linear scaling. Note that because of the logarithmic scale, value 0 cannot be displayed. Therefore values $< 0.001$ are displayed as 0.001. We chose this value because maximum relative error $\max_{v\in V}(\frac{c_C^{approx}(v)}{c_C^{exact}(v)})$ is at $\approx 1000$ in our experiments.

Errors for important vertices are small and the advantage of the bisection method and linear scaling compared to Brandes' method is small. But for unimportant vertices especially the bisection method is much better than other methods despite the fact that it uses fewer pivots because it needs more time to calculate contributions for one pivot. For linear scaling the advantage over Brandes' method is smaller. Plots of relative error distinguished by more levels can be found in the appendix section 5.

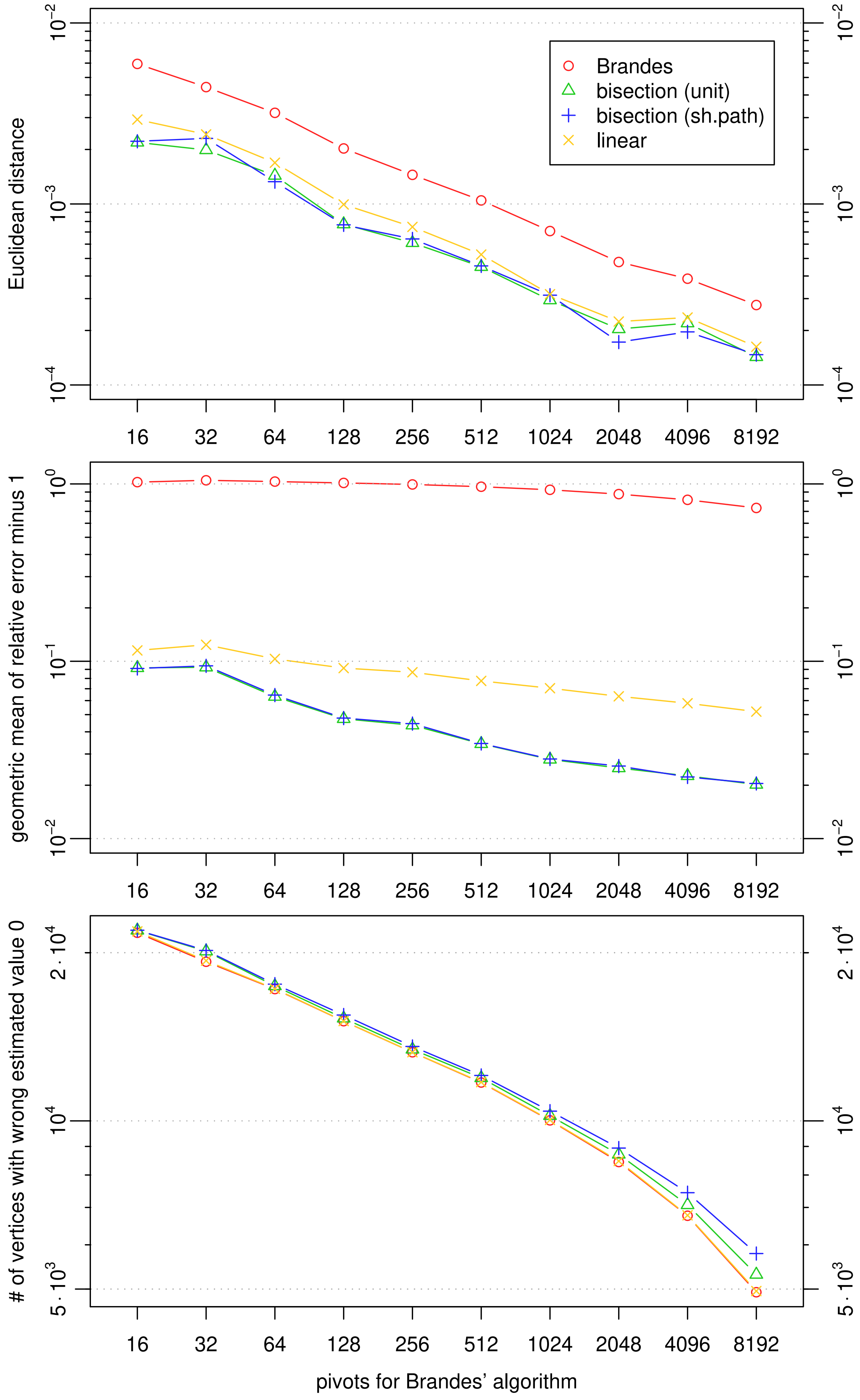


Figure 8: Global scores for canonical betweenness centrality estimation

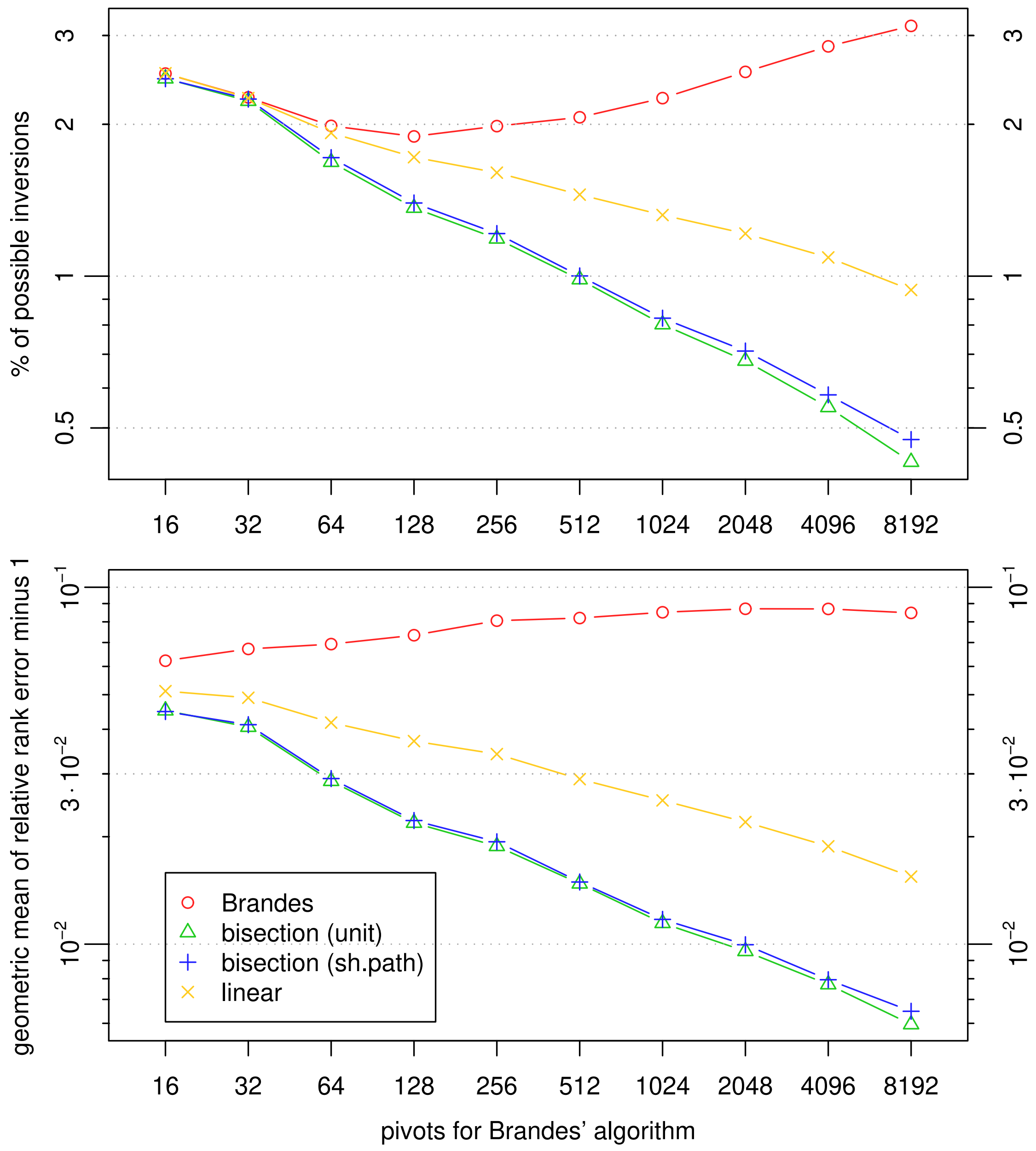


Figure 9: Global scores for canonical betweenness centrality estimation

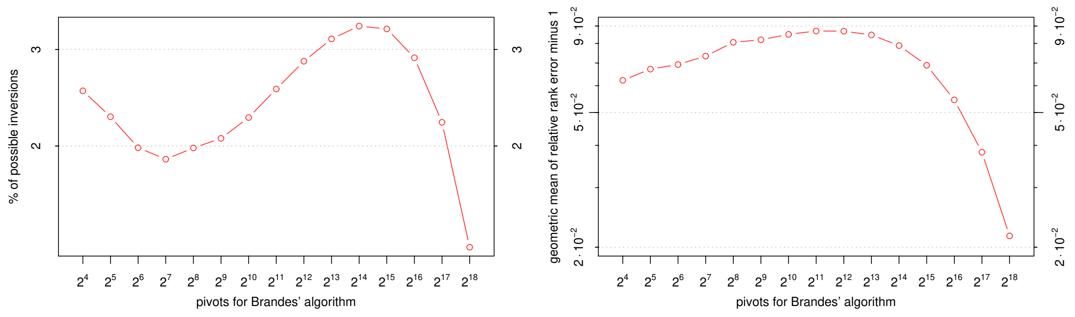


Figure 10: Global scores for canonical betweenness centrality estimation, Brandes' method

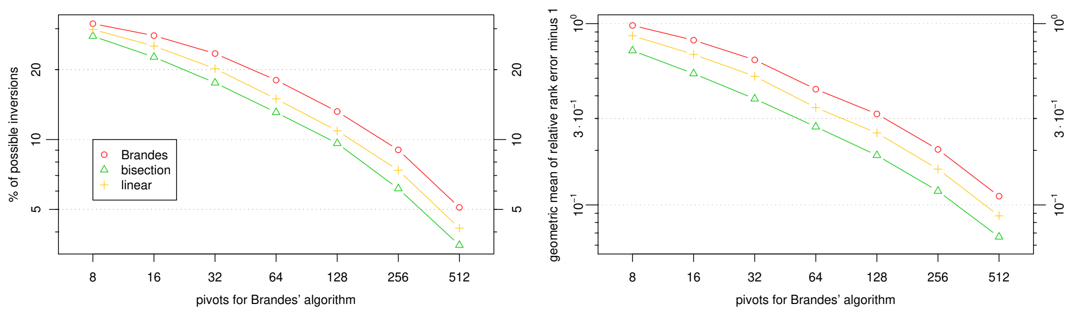


Figure 11: Global scores for canonical betweenness centrality estimation, random graph, unweighted, undirected, 1 000 vertices, 10 074 edges

Ranks can be used for example to construct overlay graphs for highway-node routing. In Figure 13 *rank errors*

$$\frac{r^{approx}(v)}{r^{exact}(v)}$$

are compared.

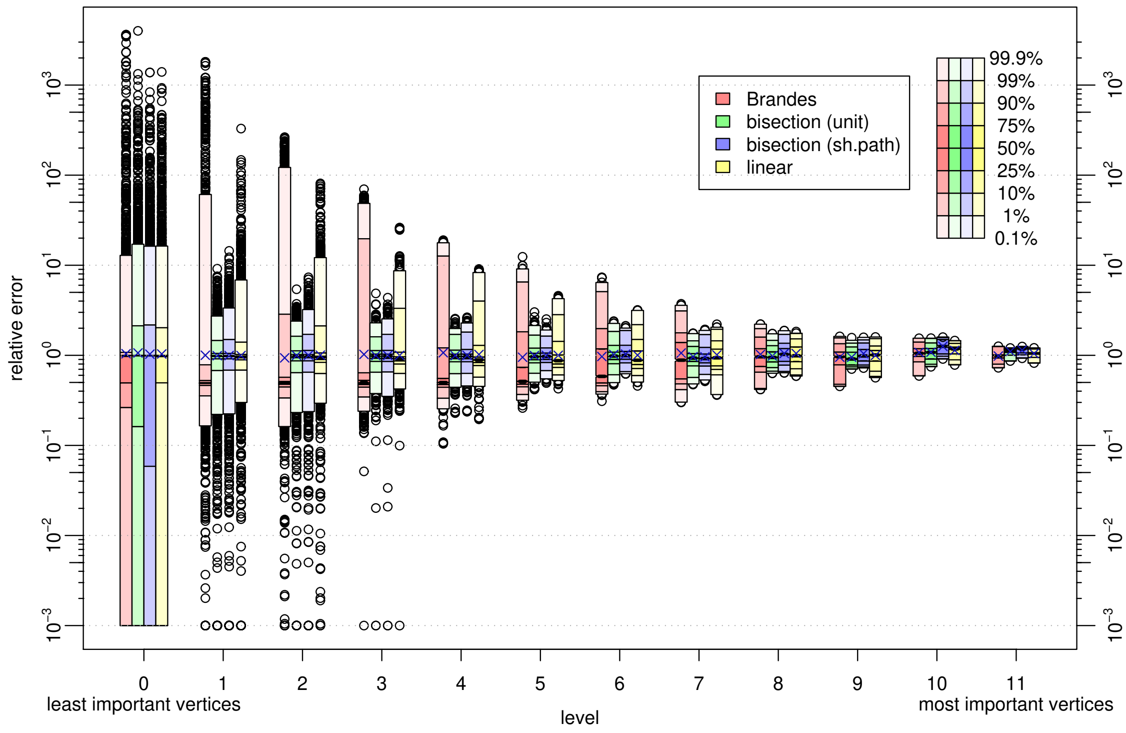
Looking at the most important vertices, grouped in levels 10 and 11, Brandes' method can compete with linear scaling and it looks even better than bisection method with unit distance, at least for 32 pivots. But with 128 pivots bisection method is the best method. The large error of bisection method with 32 pivots can be explained with the ranking system. Because ranks of important vertices are small, even a small rank error can result in a large relative rank error. Note that upper and lower quartiles for bisection method are better than for Brandes' method.

The first major result of this thesis is that linear scaling and especially the bisection method are better than Brandes' method. We presented our results solely for the Belgium graph but we tested other real-world networks and observed similar results.

The bisection method with unit distance and with shortest path distance show similar results. Even though the bisection method with shortest path distance needs more time per pivot and thus having fewer pivots calculated than the bisection method with unit distance, it has smaller rank errors.

In Figure 12 you note that the median of the Brandes' method is at 0.5 and for the bisection method and linear scaling it is at 1. This phenomenon is not a result of an error in our implementation of Brandes' algorithm. The mean value is still at 1 and this phenomenon alleviates as the number of pivots grows, see Figure 14. For level 0 this phenomenon is not that distinct because there are many vertices $v$ with $c_C(v) = 0$ which are always estimated correctly.

Notable are also large outliers especially in low levels. This is mainly a result of high variance of contributions of different pivots. In Figure 15 the frequency of different contributions for an exemplary outlier is plotted. Brandes' method has some huge contributions leading to large errors whereas linear scaling reduces this contributions and the bisection method has almost no variance.



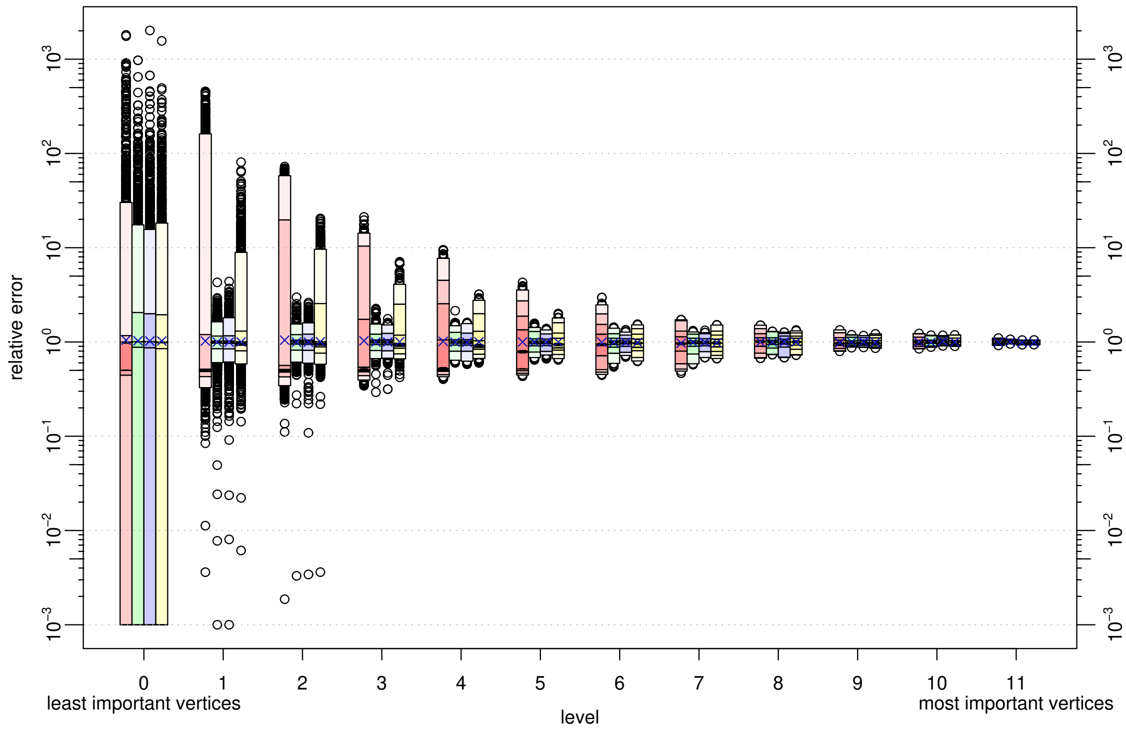
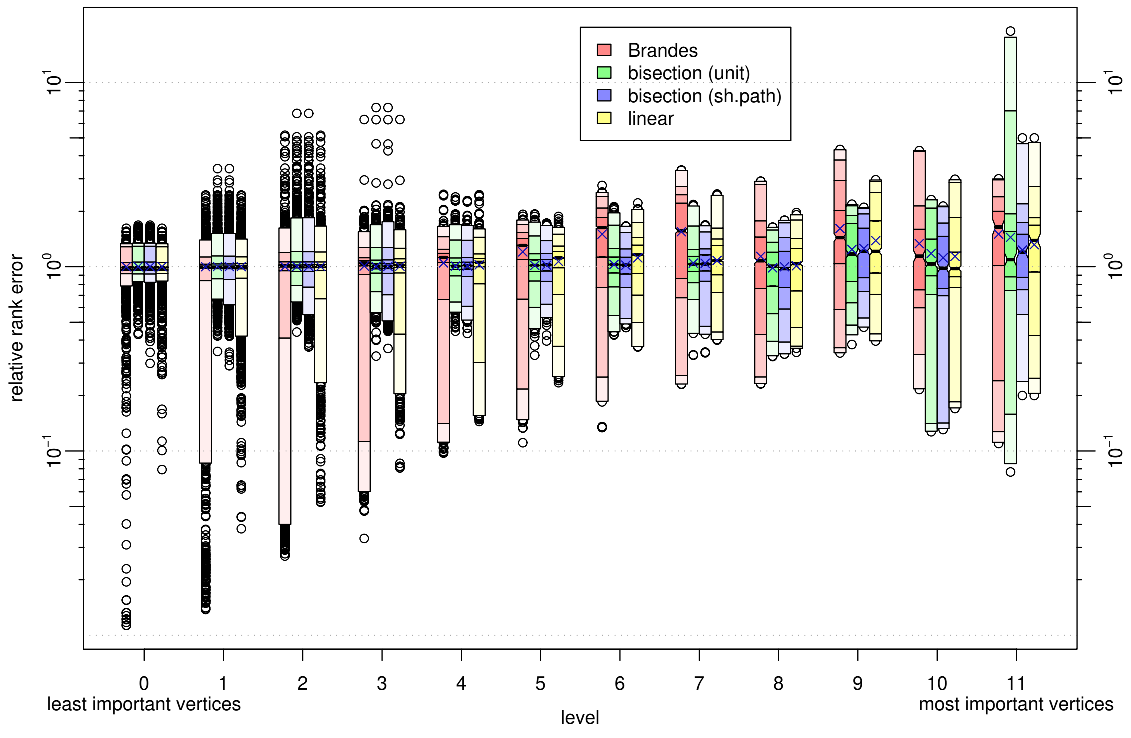


Figure 12: Relative error for canonical betweenness centrality estimation, classified by level, 32 pivots above, 128 below, boxes represent 0.1%, 1%, 10%, 25%, 50%, 75%, 90%, 99%, 99.9% quantiles, blue cross is mean value, circles are outliers



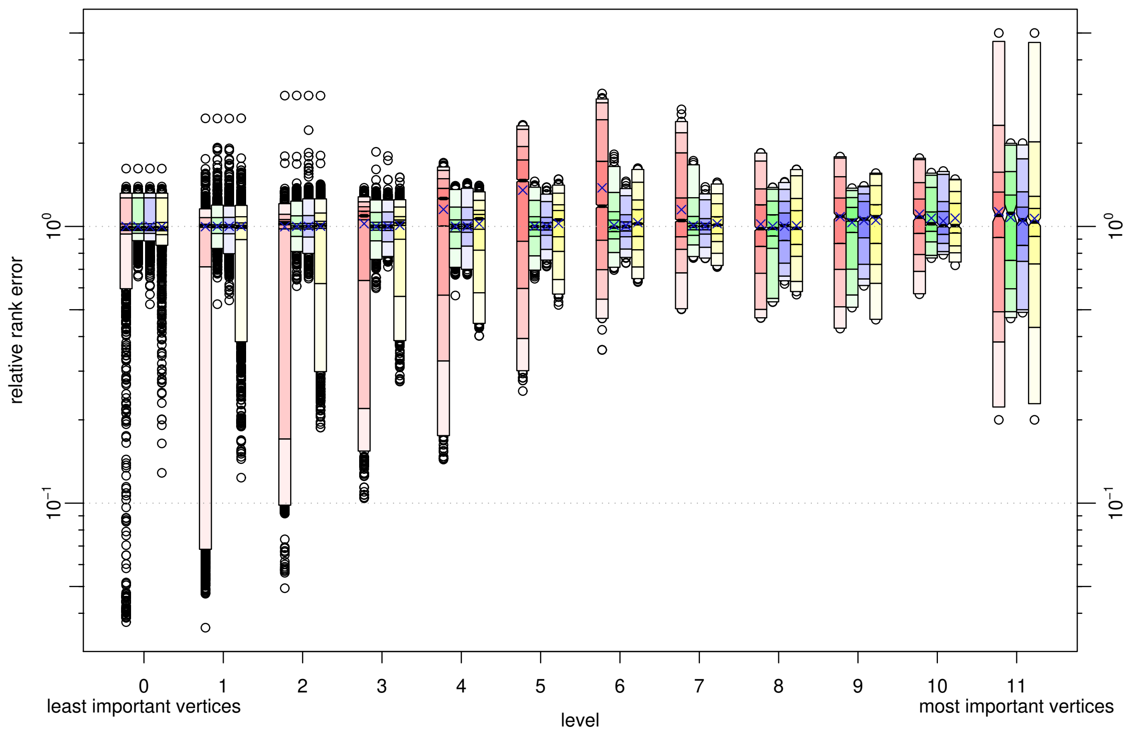


Figure 13: Relative rank error classified by level, 32 pivots above, 128 pivots below, boxes represent 0.1%, 1%, 10%, 25%, 50%, 75%, 90%, 99%, 99.9% quantiles, blue cross is mean value, circles are outliers

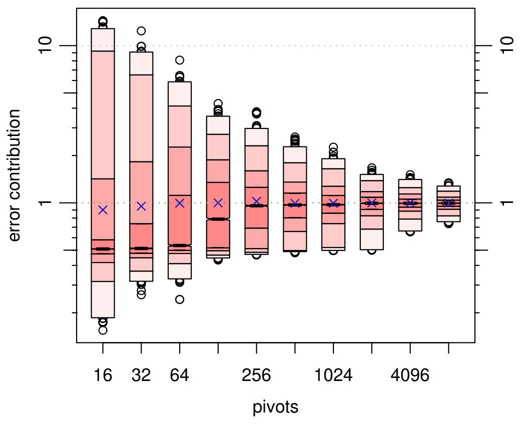


Figure 14: Brandes' method for level 5, boxes represent 0.1%, 1%, 10%, 25%, 50%, 75%, 90%, 99%, 99.9% quantiles, blue cross is mean value, circles are outliers

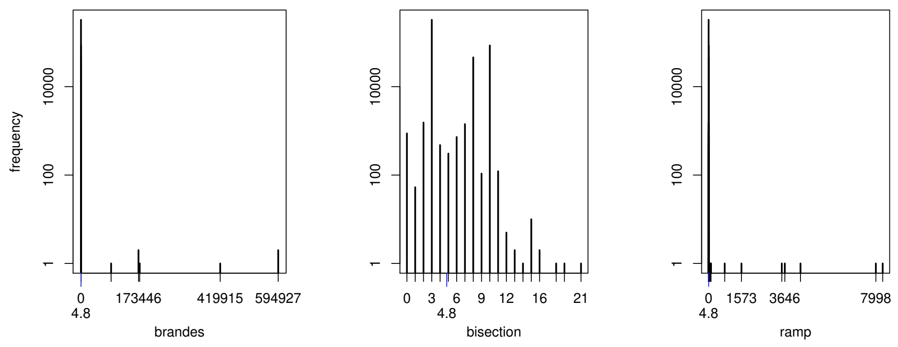


Figure 15: Variance of contributions of an outlier vertex. Horizontal are all possible contributions and vertical is frequency of the contribution. The small blue line on horizontal axis indicates $\frac{c_C(v)}{n}$.

## Problems of Linear Scaling

The linear approach is easy to implement, quite fast and shows better results than Brandes' method. But in large graphs, vertices with small canonical betweenness centrality value are estimated worse than with the bisection method. The reason is that vertices near a pivot receive high contributions to their counter value and linear reduction softens this phenomenon only a little bit. Let us start with a thought experiment. Think of a uniform grid like in Figure 16, a very simple model of a street graph. Most shortest paths from the upper quarter, bounded by the two blue lines, will go through vertex $v$. Draw a square with edge length $r$ around $s$. Each vertex on the edges of the square has $\Theta(r)$ distance to $s$, there are $\approx \frac{r^2-(r-1)^2}{4} = \frac{2r-1}{4}$ vertices on the upper edge of the square. Each of these vertices contributes $\Theta(\frac{1}{r})$ to the estimated counter value $c_C^{approx}(v)$. This is about $\Theta(1)$ for all vertices on the quarter circle and for a graph with edge length $R$, $\Theta(R)$ in total. Hence the contribution of pivot $s$ to the canonical betweenness centrality value $c_C(v)$ of a pivot $v$ close to $s$ partially depends on $R$. In large graphs it is likely that $c_C(v)$ is overestimated.

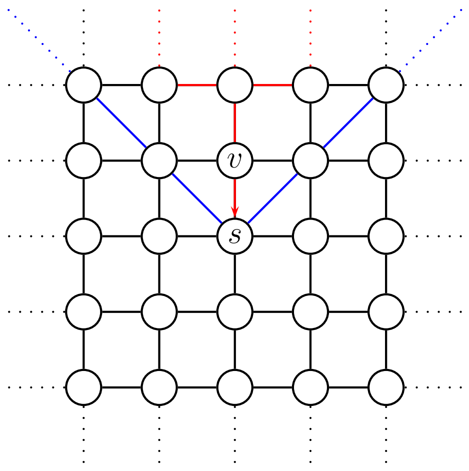


Figure 16: Example for linear scaling with $c_C^{approx}(v) \in \Omega(\text{edge length})$ for each vertex $v$

This is a partial explanation for the worse performance compared to the bisection method. This *tree-effect* can be observed in real-world networks, too. Figure 17 shows a part of Spain. Calculation of 16 pivots forward and backward are compared to our best values of Western Europe, an approximation with 8192 pivots using the bisection method. We did not use exact value because calculation would take $\approx$ 12 years. Using the bisection method may favor the results for the bisection method, but only a little because of the huge difference between 16 and 8192 pivots. The blue diamond is a backward pivot, cyan dots are underestimated by more than factor $10^{-8}$, red dots are overestimated, the darker the red the lower is the overestimation. Brandes' method has many bright red dots near the pivot whereas linear scaling has darker dots with smaller error. But the dots get a little brighter as they reach the pivot. That phenomenon corresponds to the one described in Figure 16.

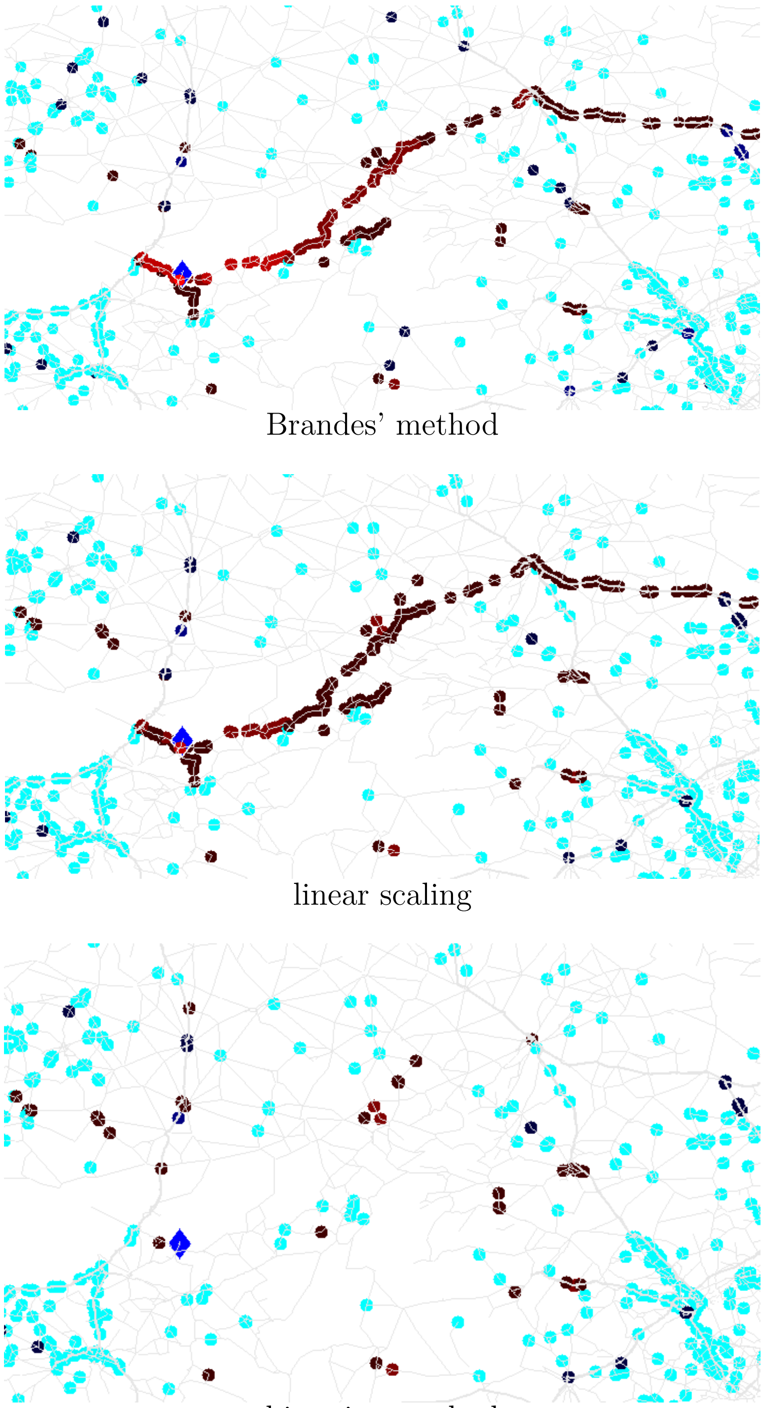


Figure 17: Tree-effect. A part of the road network of Spain, 16 pivots compared to 8192 pivots with the bisection method, the blue diamond is a backward pivot, cyan dots are underestimated by more than factor $10^{-8}$, red dots are overestimated, the darker the red the lower is the overestimation.

### Problems of the Bisection Method

Although the bisection method seems to be the best choice to estimate canonical betweenness centrality there are still many outliers especially in low levels, these are the cyan dots in Figure 17. Affected are mostly vertices $v$ with small canonical betweenness centrality value $c_C(v)$ and only few *contributors*, pivots $s$ with $\delta_{s\bullet}(v) > 0$. Three exemplary vertices are shown in Figure 18. It is highly likely that none of the chosen pivots is a contributor and such a vertex gets canonical betweenness centrality value zero. On the other hand, it can happen that a single contributor is among a small number $k$ of pivots. Through extrapolation with $k \ll n$, the estimated canonical betweenness centrality value is too large.

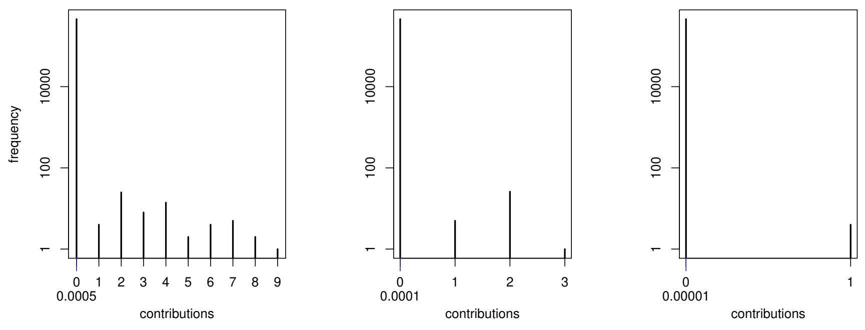


Figure 18: Variance of contributions of three outlier vertices. Horizontal are all possible contributions and vertical is the frequency of the contributions. The small blue line on the horizontal axis indicates $\frac{c_C(v)}{n}$.

# 3 Approximation of General Betweenness Centrality

General betweenness centrality can be seen as the *multipath variant* of canonical betweenness centrality because it respects all shortest paths between pairs of vertices. We will adopt the framework of canonical betweenness centrality but need to modify it to work with multiple shortest paths. Brandes' method and linear scaling fit in well but the bisection method needs to be adapted to obtain an efficient implementation.

## 3.1 Methods

Because multiple shortest paths between two vertices $s, t \in V$ can occur, unit distance does not only depend on $s$ and $t$ but also on the shortest path currently processed. So a modification of the $d$ function of Section 2.1 is necessary. Let $P \in SP_{st}$ be a shortest path between $s, t \in V$ represented by its vertices and a strict order $\prec_P$ with $s \prec_P t$ indicating the sequence of vertices on the shortest path. Let $d_P : P \times P \to \mathbb{R}^{\geq 0}$ be a function that satisfies

$$\forall u, v, w \in P : s \preceq_P u \prec_P v \prec_P w \preceq_P t \Rightarrow d_P(u, w) = d_P(u, v) + d_P(v, w) \tag{22}$$

With this definition unit distance can be used. Definitions $f_\ell$, $\bar{f}_\ell$ can be adopted from Section 2.1 but we introduce a different notation for improved readability. We need to weight contribution depending on $\ell = d_P(s, t)$.

$$f_{s,t,P} := f_{d_P(s,t)} \tag{23}$$

$$\bar{f}_{s,t,P} := \bar{f}_{d_P(s,t)} \tag{24}$$

To adapt contributions for forward and backward pivots to the new $d_P$ function, define for $s, t, v \in V$

$$\delta_{s\bullet}(v) := \sum_{t \in V} \frac{\sum\limits_{P \in SP_{st}(v)} f_{s,t,P}(d_P(s, v))}{\sigma_{st}} \tag{25}$$

$$\delta_{\bullet t}(v) := \sum_{s \in V} \frac{\sum\limits_{P \in SP_{st}(v)} \bar{f}_{s,t,P}(d_P(v, t))}{\sigma_{st}} \tag{26}$$

$\delta_{s\bullet}(v)$ and $\delta_{\bullet t}(v)$ are contributions to the general betweenness centrality value of $v$ for a forward pivot $s \in V$ or a backward pivot $t \in V$. Define a random variable with equal distribution among all $2n$ possible events. Analogous to (11),

$$X = \begin{cases} 2n \cdot \delta_{s\bullet}(v) & \text{if forward pivot } s \text{ is chosen} \\ 2n \cdot \delta_{\bullet t}(v) & \text{if backward pivot } t \text{ is chosen} \end{cases} \tag{27}$$

**Lemma 2** *$X$ is an unbiased estimator meaning $\forall v \in V : E(X) = c_B(v)$ for all methods constructed satisfying conditions listed above.*

*Proof.*

$$\begin{aligned}
E(X) &\overset{(27)}{=} \frac{1}{2n}\left(\sum_{s\in V}(2n\cdot\delta_{s\bullet}(v)) + \sum_{t\in V}(2n\cdot\delta_{\bullet t}(v))\right)\\
&\overset{(25),(26)}{=} \sum_{s\in V}\sum_{t\in V}\frac{\sum\limits_{P\in SP_{st}(v)} f_{s,t,P}(d_P(s,v))}{\sigma_{st}} + \sum_{t\in V}\sum_{s\in V}\frac{\sum\limits_{P\in SP_{st}(v)} \bar{f}_{s,t,P}(d_P(v,t))}{\sigma_{st}}\\
&= \sum_{s,t\in V}\frac{\sum\limits_{P\in SP_{st}(v)}\left(f_{s,t,P}(d_P(s,v)) + \bar{f}_{s,t,P}(d_P(v,t))\right)}{\sigma_{st}}\\
&\overset{(22),(8)}{=} \sum_{s,t\in V}\frac{\sum\limits_{P\in SP_{st}(v)} 1}{\sigma_{st}} = \sum_{s,t\in V}\frac{\sigma_{st}(v)}{\sigma_{st}}\\
&\overset{(1)}{=} c_B(v)
\end{aligned}$$

□

### Brandes' Method

Brandes' method does not need the above created complex design. Definitions of $f_\ell$ (12) and $\bar{f}_\ell$ (13) still apply and no $d_P$ function is required.

### Linear Scaling

Linear scaling needs the same adaptations as Brandes' method to work with general betweenness centrality instead of canonical betweenness centrality. Definitions (14) and (15) still apply.

### Bisection Method

The bisection method is the reason why we introduced the $d_P$ function. In common cases unit distance can only be defined on an a distinct path. The unit distance on a path with few long edges is different than the unit distance on a path with many short edges, although both paths have the same length. Functions $f_\ell$ (16) and $\bar{f}_\ell$ (17) are still the same as for canonical betweenness centrality. If we used shortest path distance instead of unit distance the complex design would not be necessary.

The bisection method for general betweenness centrality implemented with the same algorithm as Canonical-Bisection has time requirements not in $O(SSSP) + O(n)$ because there could be too many shortest paths.

$$\sum_{s,t\in V} |SP_{st}| \in \omega(n) \tag{28}$$

For example in a grid like in Figure 16, two vertices $s, t \in V$ with horizontal distance $a$ and vertical distance $b$ have $|SP_{st}| = \binom{a+b}{a}$. Having a $r \times r$ grid and looking at vertices $s$ in top left corner and $t$ in bottom right corner yields an exponential number of shortest paths.

$$|SP_{st}| = \binom{2r}{r} \in \Omega(2^r) \tag{29}$$

Therefore approximation of general betweenness centrality with the bisection method can be time-consuming. Because we compare all algorithms by runtime, we use another approach to add a variant of the bisection method and compare it to the pure bisection method. The variant uses sampling of shortest paths. For each sample only one shortest path between each pair of vertices is taken into account. For each vertex $s \in V$ all shortest path with source $s$ must form a tree so they can be processed efficiently.

We introduce a notation for subpaths of a path $P \in SP_{st},\ s,t \in V$. For $v \in V$, $s \preceq_P v$ let $P|_{s\to v}$ denote the subpath from $s$ to $v$. Let $h$ be the number of samples. Let $P^i_{st} \in SP_{st}$ be randomly chosen satisfying

$$\forall s, t_1, t_2, w \in V : w \in P^i_{st_1} \cap P^i_{st_2} \Rightarrow P^i_{st_1}|_{s\to w} = P^i_{st_2}|_{s\to w} \tag{30}$$

For $s \in V$, $i \in \{1, \ldots, h\}$ fixed, if two chosen shortest paths have a vertex $w$ in common, the subpath between $s$ an $w$ must be the same leading to a shortest path tree with source $s$.

A sample $i$ contributes to $c_B(v)$ if $v \in q^i_{st}$. To define the contribution for a forward and a backward pivot another mathematical definition is necessary.

$$[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{if } x \text{ is false} \end{cases}$$

With this mathematical tool the contributions for a forward pivot $\delta_{s\bullet}(v)$ and for a backward pivot $\delta_{\bullet t}(v)$ are as follows.

$$\delta_{s\bullet}(v) := \sum_{t\in V} \left( \frac{1}{h} \cdot \sum_{i=1}^{h} \left( \left[ v \in q^i_{st} \right] f_{s,t,q^i_{st}}(d_{q^i_{st}}(s,v)) \right) \right) \tag{31}$$

$$\delta_{\bullet t}(v) := \sum_{s\in V} \left( \frac{1}{h} \cdot \sum_{i=1}^{h} \left( \left[ v \in q^i_{st} \right] \bar{f}_{s,t,q^i_{st}}(d_{q^i_{st}}(v,t)) \right) \right) \tag{32}$$

Sampling limits the number of visited shortest paths resulting in an algorithm with time requirements in $O(SSSP)+O(hn)$. Note the divisor $h$ instead of $\sigma_{st}$ because if there is a shortest path between two vertices, exactly $h$ not necessarily distinct paths are taken into account. All regarded shortest paths between a pair of vertices should contribute at most 1 to the general betweenness centrality value of a vertex $v$. We use the same extrapolation as above but Lemma 2 cannot be applied. To find an unbiased estimator we need to have an uniform distribution among all $P \in SP_{st}$. We describe how we select $P \in SP_{st}$ and we find an uniform distribution. All shortest path with source $s$ form an acyclic graph. We form a shortest path tree out of this acyclic search graph. Each vertex can have multiple parents. If we reduce the number of parents to one, we will gain a tree. To get a unform distribution among all shortest paths, a single parent $p$ of $t$ is chosen randomly with weight $\sigma_{sp}$, i.e. with probability $\frac{\sigma_{sp}}{\sigma_{st}}$.

**Lemma 3** *$P^i_{st}$ is chosen in $SP_{st}$ with uniform distribution.*

*Proof.* Induction for $\sigma_{st}$:

$\sigma_{st} \in \{0,1\}$ : $\checkmark$

$\sigma_{st} \geq 2$ : $prob(\text{parent p is chosen}) = \frac{\sigma_{sp}}{\sigma_{st}}, prob(P^i_{sp} \in SP_{sp} \text{ is chosen}) = \frac{1}{\sigma_{sp}}$ (IH)

$\Rightarrow prob(P^i_{st} \in SP_{st} \text{ is chosen}) = \frac{\sigma_{sp}}{\sigma_{st}} \cdot \frac{1}{\sigma_{sp}} = \frac{1}{\sigma_{st}}$

□

Now it is only a small step to see $E(X) = c_B(X)$. Because $\frac{1}{h} \cdot E(h \cdot X) = E(X)$, only the case $h = 1$ needs to be considered. We forget index $i$ and only choose paths $P_{st}$. Define another random variable $Y_{st}$ to represent $[v \in P_{st}]$.

$$Y_{st} = \begin{cases} 1 & \text{if } v \in P_{st} \\ 0 & \text{if } v \notin P_{st} \end{cases} \tag{33}$$

The definition of $\sigma_{st}$, $\sigma_{st}(v)$ and Lemma 3 yield

$$E(Y_{st}) = \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{34}$$

Now it is possible to prove that random variable $X$ is unbiased.

**Lemma 4** *$E(X) = c_B(v)$ for the bisection sampling method*

*Proof.*

$$\begin{aligned}
E(X) &= \frac{1}{2n} \cdot \left( \sum_{s \in V} 2n \cdot \delta_{s\bullet}(v) + \sum_{t \in V} 2n \cdot \delta_{\bullet t}(v) \right) \\
&\overset{(31),(32)}{=} \sum_{s \in V} \sum_{t \in V} E\left(Y_{st} \cdot f_{s,t,P_{st}}(d_{P_{st}}(s,v))\right) + \sum_{t \in V} \sum_{s \in V} E\left(Y_{st} \cdot \bar{f}_{s,t,P_{st}}(d_{P_{st}}(v,t))\right) \\
&= \sum_{s,t \in V} E\left(Y_{st}\left(f_{s,t,P_{st}}(d_{P_{st}}(s,v)) + \bar{f}_{s,t,P_{st}}(d_{P_{st}}(v,t))\right)\right) \\
&\overset{(22),(8)}{=} \sum_{s,t \in V} E(Y_{st}) \\
&\overset{(34)}{=} \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \\
&\overset{(1)}{=} c_B(v)
\end{aligned}$$

□

## Error bounds

Error bounds of canonical betweenness centrality approximation also apply to general betweenness centrality approximation. But we need to treat the bisection sampling method differently. All shortest paths between $s, t \in V$ can contribute to $c_B(v)$ by at most one since $\frac{\sigma_{st}(v)}{\sigma_{st}} \leq 1$. Because the definitions of $f_\ell$ and $\bar{f}_\ell$ have not been changed, $M = 2n(n-2) \cdot \alpha$ is still an upper bound leading to the same error bounds.

Bisection sampling could be interpreted in two different ways, only the first one is valid. Either see the contribution of one pivot as an evaluation of a random variable. Then $M$ with $\alpha = 1$ is an upper bound. As the upper bound is independent of the number of samples even the bisection sampling method with one sample has the error bounds proven in Section 2.

Or one could see each pivot and each sample as an distinct evaluation of a random variable. Then it would have same upper bound $M$ but more evaluations. But those evaluations would not be independent because all $h$ samples are from the same pivot, and Hoeffdings inequality (18) cannot be applied.

## 3.2 Algorithms

The algorithms for canonical betweenness centrality approximation only need small modifications to estimate general betweenness centrality. First we will present Brandes' algorithm and then introduce the bisection algorithm and the bisection sampling algorithm.

### Brandes' Algorithm

Brandes modifies the breath first search to cope with multiple shortest paths. It is necessary to store the number of shortest paths and multiple parents per vertex. That means the result is an acyclic graph with distinct root. Same modifications are needed for Dijkstra's algorithm in weighted graphs but they do not affect asymptotic time requirements. Because more than one shortest path is possible the recursion equation (21) is slightly different. Now the values of a child vertex $w$ need to be weighted with fraction $\frac{\sigma_{sv}}{\sigma_{sw}}$ of shortest path that lead to $w$ via $v$.

$$\delta_{s\bullet}(v) = \sum_{w \in C_s(v)} \frac{\sigma_{sv}}{\sigma_{sw}} \left(1 + \delta_{s\bullet}(w)\right) \tag{35}$$

Brandes' algorithm differs, besides MultipathSSSP (Line 5) and multiple parents (Line 10), only by this factor $\frac{\sigma_{sv}}{\sigma_{sw}}$ (Line 11), see Algorithm 4.

**Algorithm 4**: Betweenness-Brandes

```
Input: Graph $G = (V, E)$ with $n = |V|$, number of pivots $k$
Output: Vertex-array $c$ of estimated general betweenness centrality values for each $v \in V$
1  foreach $v \in V$ do
2      $c[v] \leftarrow 0$;
3  for $i = 1$ to $k$ do
4      Choose $s \in V$ and search direction uniformly at random;
5      Solve MultipathSSSP($s$,direction);
6      foreach $v \in V$ do
7          $c_{local}[v] \leftarrow 0$;
8      forall $w \in V$ reachable from $s$, $w \neq s$ in order of non-increasing distance to $s$ do
9          $c[w] = c[w] + c_{local}[w]$;
10         forall parents $v$ of $w$ do
11             $c_{local}[v] \leftarrow c_{local}[v] + \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (c_{local}[w] + 1)$;
   // extrapolate
12 foreach $v \in V$ do
13     $c[v] \leftarrow c[v] \cdot \frac{2n}{k}$
14 return $c$
```

### Linear Scaling Algorithm

The linear scaling algorithm is modified like Brandes' algorithm: MultipathSSSP and the factor $\frac{\sigma_{sv}}{\sigma_{sw}}$ are changed. Since the modifications to Algorithm 2 are so small, we omit the pseudocode of the Betweenness-Linear-Scaling algorithm. The recursion equation of Brandes' algorithm (35) needs a small change to work for the linear scaling algorithm. Because of linear scaling contributions of a shortest path between $s$ and $t$ to $c_B(v)$ are weighted with $\frac{d(s,v)}{d(s,t)}$. The numerator depends on $v$ while the denominator does not. The trick is to add up $\frac{1}{d(s,t)}$ and multiply with $d(s,v)$ at the end.

$$\delta_{s\bullet}(v) = d(s,v) \cdot \sum_{w \in C_s(v)} \left( \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{1 + \delta_{s\bullet}(w)}{d(s,w)} \right) \tag{36}$$

$c_{local}[v] = \frac{\delta_{s\bullet}(v)}{d(s,v)}$ still holds like in Section 2.2.

### Bisection Algorithm

The bisection algorithm processes each shortest path from source $s$ and adds up fractions $\frac{1}{\sigma_{sv}}$ instead of 1 for each shortest paths between $s$ and $v$, see Algorithm 5 Line 27. Also the former data structure with *first child* and *next sibling* cannot be used any longer because there can be multiple parents and the *next sibling* value depends on the parent. An array of children is used instead.

Recursion equation (35) cannot be applied as each vertex $v$ is visited $\sigma_{sv}$ times by the loop starting at Line 9. Brandes' algorithm only visits each vertex once. To prove that the Betweenness-Bisection algorithm implements the bisection method, another recursion equation is necessary. We will only present this equation for the forward direction. The backward direction is analogous. For $s, t, v \in V, P \in SP_{sv}$ let $\sigma_{st}(v, P)$ be the number of shortest paths between $s$ and $t$ with path prefix $P$. The connection to $\sigma_{st}$ is simply

$$\sigma_{st}(v) = \sum_{P \in SP_{sv}} \sigma_{st}(v, P)$$

because all shortest path from $s$ via $v$ have a prefix that is a shortest path between $s$ and $v$. The contribution of a pivot $s$ and path prefix $P$ is then

$$\delta_{s\bullet}(v, P) := \sum_{t \in V} \frac{\sigma_{st}(v, P)}{\sigma_{st}} \tag{37}$$

leading to the contribution of a pivot $s$:

$$\delta_{s\bullet}(v) = \sum_{P \in SP_{sv}} \delta_{s\bullet}(v, P) \tag{38}$$

To get a recursion with all children $w$ of $v$ we need to add up all contributions along shortest paths $Q \in SP_{sw}$ that have the path $P \in SP_{sv}$ as prefix.

$$\delta_{s\bullet}(v, P) \overset{(*)}{=} \sum_{w \in C_s(v)} \left( \frac{\sigma_{sw}(v, P)}{\sigma_{sw}} + \delta_{s\bullet}(w, P) \right) \overset{(**)}{=} \sum_{w \in C_s(v)} \sum_{Q \in SP_{sw}, Q|_{s \to v} = P} \left( \frac{1}{\sigma_{sw}} + \delta_{s\bullet}(w, Q) \right) \tag{39}$$

$(*)$: the fractions of shortest paths with prefix $P$ are the fraction of shortest paths from $s$ to $w \in C_s(v)$ via $v$ $(= \frac{\sigma_{sw}(v,P)}{\sigma_{sw}})$ plus the fractions of shortest paths $w$ lies on $(= \delta_{s\bullet}(w, P))$.
$(**)$: all different shortest paths $Q$ from $s$ to $w$ with prefix $P$ are added up separately. If there is only one edge between $v$ and $w$, there will only be one path. But our definition of a graph allows multiple edges, leading to the above equation.

In Algorithm 5 the stack $S$ represents the path from $s$ to $w$. In Line 23 identify $a = \delta_{s\bullet}(w, S)$. This line is executed for each different shortest path $S \in SP_{sw}$ resulting in

$$c[w] = \sum_{P \in SP_{sw}} \delta_{s\bullet}(w, S) \overset{(38)}{=} \delta_{s\bullet}(w)$$

**Algorithm 5**: Betweenness-Bisection

```
Input: Graph $G = (V, E)$ with $n = |V|$, number of pivots $k$
Output: Vertex-array $c$ of estimated general betweenness centrality values for each $v \in V$
1  foreach $v \in V$ do
2      $c[v] \leftarrow 0$
3  for $i$ = 1 to $k$ do
4      Choose $s \in V$ and search direction uniformly at random;
5      Solve MultipathSSSPWithChildren($s$,$direction$);
6      $S \leftarrow \emptyset$ ; // stack with random access, items (vertex,counter)
7      $C \leftarrow \emptyset$ ; // stack of children
8      push($S$,$(s, 0)$);
9      while $S \neq \emptyset$ do
           // count for vertex in the middle only on forward search
10         if forward search then
11             if size($S$)$= 1$ then $m \leftarrow 0$ else $m \leftarrow \left\lfloor \frac{\texttt{size}(S)-2}{2} \right\rfloor$;
12         else
13             $m \leftarrow \left\lfloor \frac{\texttt{size}(S)-1}{2} \right\rfloor$;
14         $(v, a) \leftarrow S[m]$;
15         $S[m] \leftarrow (v, a - \frac{1}{\sigma_{sv}})$; // 'bisection trick'
16         if back($S$) has children then
17             push($C$,all children of back($S$));
18             $v \leftarrow$ pop($C$);
19             push($S$,$(v, 0)$);
20         else
               // remove vertices from stack with no children left and increment
                  counters
21             while $S \neq \emptyset$ and back($S$) has no children left on $C$ do
22                 $(w, a) \leftarrow$back($S$);
23                 $c[w] \leftarrow c[w] + a$;
24                 pop($S$);
25                 if $S \neq \emptyset$ then
26                     $(v, b) \leftarrow$back($S$);
27                     back($S$)$\leftarrow (v, b + a + \frac{1}{\sigma_{sw}})$;
28             if back($S$) has children left on $C$ then
29                 $w \leftarrow$ pop($C$);
30                 push($S$,$(w, 0)$);
   // extrapolate
31 foreach $v \in V$ do
32     $c[v] \leftarrow c[v] \cdot \frac{2n}{k}$
33 return $c$
```

In Line 27 identify $a + \frac{1}{\sigma_{sw}} = \delta_{s\bullet}(w, S) + \frac{1}{\sigma_{sw}}$. This recursion matches equation (39).

Sampling is implemented by forming a tree out of the resulting acyclic graph of MultipathSSSP. For each vertex $v$, only one parent $p$ is randomly chosen with probability $\frac{\sigma_{sp}}{\sigma_{sv}}$. Our implementation randomly chooses an integer $x$ between 1 and $\sigma_{sv}$. Let $p_1, \ldots, p_o$ be the parents of $v$. Then $p_j$ is chosen if $x \in \left[\sum_{i=1}^{j-1} \sigma_{sp_i} + 1, \sum_{i=1}^{j} \sigma_{sp_i}\right]$. The Canonical-Bisection algorithm can be applied for each of the $h$ samples if results are divided by $h$. We do not present Betweenness-Bisection-Sampling algorithm here.

Per pivot $O(SSSP) + O(hn)$ time is required, or $O(SSSP) + O(n)$ if $h$ is fixed.

## 3.3 Experiments

The same compiler and the same hardware as in Section 2.3 are used.

To test the algorithms to approximate general betweenness centrality we focus on a different graph than for canonical betweenness centrality. The road graph of Belgium that was used there has only few multiple shortest paths and is therefore not appropriate to test general betweenness centrality. We used an Actor network based on imdb.com [14] instead, with $n = 392\,400$, $m = 16\,557\,451$. It is an unweighted undirected graph where each vertex represents an actor and for each movie there is an edge between each pair of actors that appear in the movie. Multiple edges between two actors are possible. Because this graph is unweighted, there is no difference between unit distance and shortest path distance. Therefore, we do not distinguish between the bisection method with unit distance and with shortest path distance. We compared Brandes' method to the bisection method, the bisection sampling method with 2, 4, 8 and 16 samples, and linear scaling. The definitions of error scores as in Section 2.3 now apply to general betweenness centrality $c_B(v)$. The same definition of number of pivots applies. The number of pivots refers to Brandes algorithm, all other algorithms get at most as much time as Brandes algorithm to calculate their results. Also the number of pivots specifies the number of forward pivots and the number of backward pivots, e.g. 1024 pivots indicate 1024 forward pivots and 1024 backward pivots.

Runtimes of the different algorithms can be found in Table 2. Brandes' method and linear scaling show similar time requirements. The bisection sampling runtime increases with the number of samples as expected. The bisection algorithm without sampling has a much larger runtime, we will analyze this later.

The Euclidean distance and the geometric mean of relative error in Figure 19 indicates same good results as for canonical betweenness centrality. Brandes' method compared to the bisection

| algorithm | seconds/pivot |
|---|---|
| Brandes | 6.254 |
| bisection | 29.382 |
| bisection sampling (2 samples) | 6.581 |
| bisection sampling (4 samples) | 7.014 |
| bisection sampling (8 samples) | 7.712 |
| bisection sampling (16 samples) | 9.415 |
| linear scaling | 6.284 |

Table 2: Runtime of different algorithms that approximate general betweenness centrality

sampling method with 2 samples yields an improvement of the Euclidean distance by factor $3.9 - 5.2$. Or the other way round Brandes' algorithm needs $\approx 16$ times longer to achieve the same results as the bisection sampling method with 2 samples. The bisection sampling method is better than the pure bisection method. Our experiments also suggests that two samples for bisection sampling are enough, at least for this graph. But for bisection and bisection sampling the number of vertices $v$ with $c_B^{approx}(v) = 0$ is larger than for Brandes' method and linear scaling. The bisection algorithm has theses larger numbers because time is short, it only processes 7 or 27 pivots in the time Brandes' algorithm processes 32 or 128 pivots. Interestingly it still outperforms Brandes' method. And in the case of the bisection sampling algorithm, many vertices have zero value because not all shortest paths are regarded. In both cases only vertices $v$ with small $c_C(v)$ seem to be affected because the Euclidean distance is small.

We introduce a rank $r_B(v)$ among all nodes $v \in V$ like in Section 2.3. Global scores regarding the rank are shown in Figure 20. The inversion number and the geometric mean of relative rank error show different results than Euclidean distance and geometric mean of rank error. Linear scaling is always better than Brandes' method. But both methods show show stagnating results before they head down after thousands of pivots. The bisection method and the bisection sampling method beat Brandes' method not until 1024 pivots. But they have a near-linear decrease in the double-logarithmic plot. become better with increasing number of pivots. The break point is around 1024 pivots. The bisection sampling methods benefits of more samples. In Figure 21 we compare the methods by number of pivots instead of runtime. The bisection method is in this comparison the best method. Therefore a linear time algorithm for the bisection method would make it the superior approximation method.

To further analyze the differences between the approximation methods we need to introduce levels like in Section 2.3. Figure 22 only shows results for Brandes', the bisection, bisection sampling with 2 and 8 samples and linear scaling. Especially in Level 1 and 2 many vertices are underestimated.

The bisection sampling method with 2 samples shows best results. It only underestimates unimportant nodes in Levels 0-3 to much. It seems that bisections sampling throws away to much information because there are to many different shortest path. This affects the results especially for small numbers of pivots. The bisection method often underestimates general betweenness centrality values of vertices indicated by outliers in levels 3 and 4 because it can only process few pivots during the limited runtime.

Results are not as good as for the Belgian graph with canonical betweenness centrality. But this does not mean that our approximation algorithms for general betweenness centrality are worse than our approximation algorithms for canonical betweenness centrality. First, the centrality index is different and second, the graph type is completely different. The only thing both graphs have in common that they are not synthetic.

The plot for 128 pivots indicates that more pivots yield better results indicating that our methods are unbiased and will asymptotically reach relative error 1. The same 0.5 phenomenon for Brandes' method occurs as for canonical betweenness centrality, now also slightly affecting linear scaling. Level 0 shows no error because all vertices $v$ in level 0 have $c_B(v) = 0$. Plots of relative error distinguished by more levels can be found in the appendix, Section 5.

Relative rank errors in Figure 23 indicate similar results as relative centrality value errors. Many vertices are placed to low in rank because there are too few pivots. Differences between 32 and 128 pivots indicate that more pivots will alleviate the problem. Except for the pure bisection method, Brandes' method performs worse than our presented methods.

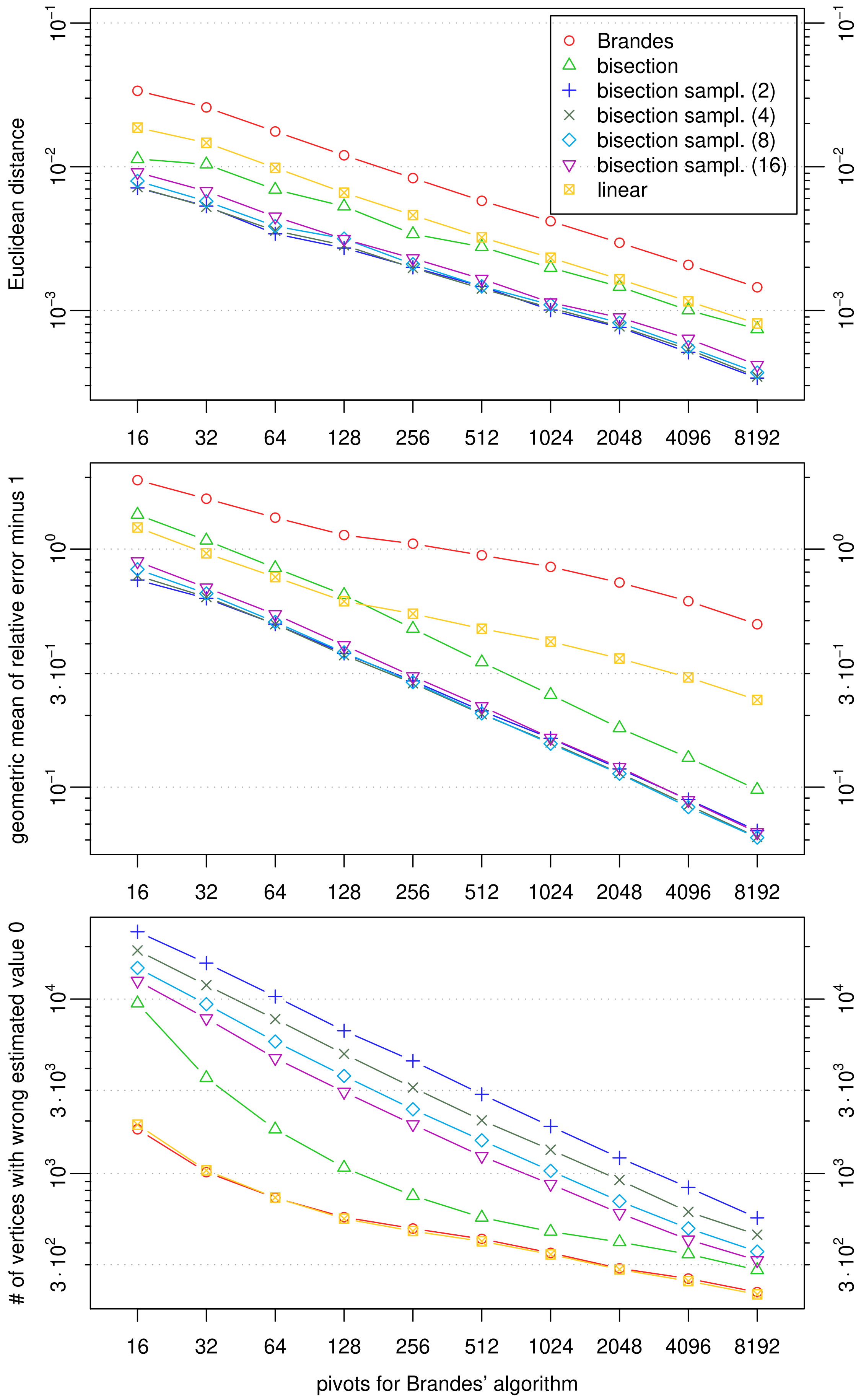


Figure 19: Global scores for general betweenness centrality estimation

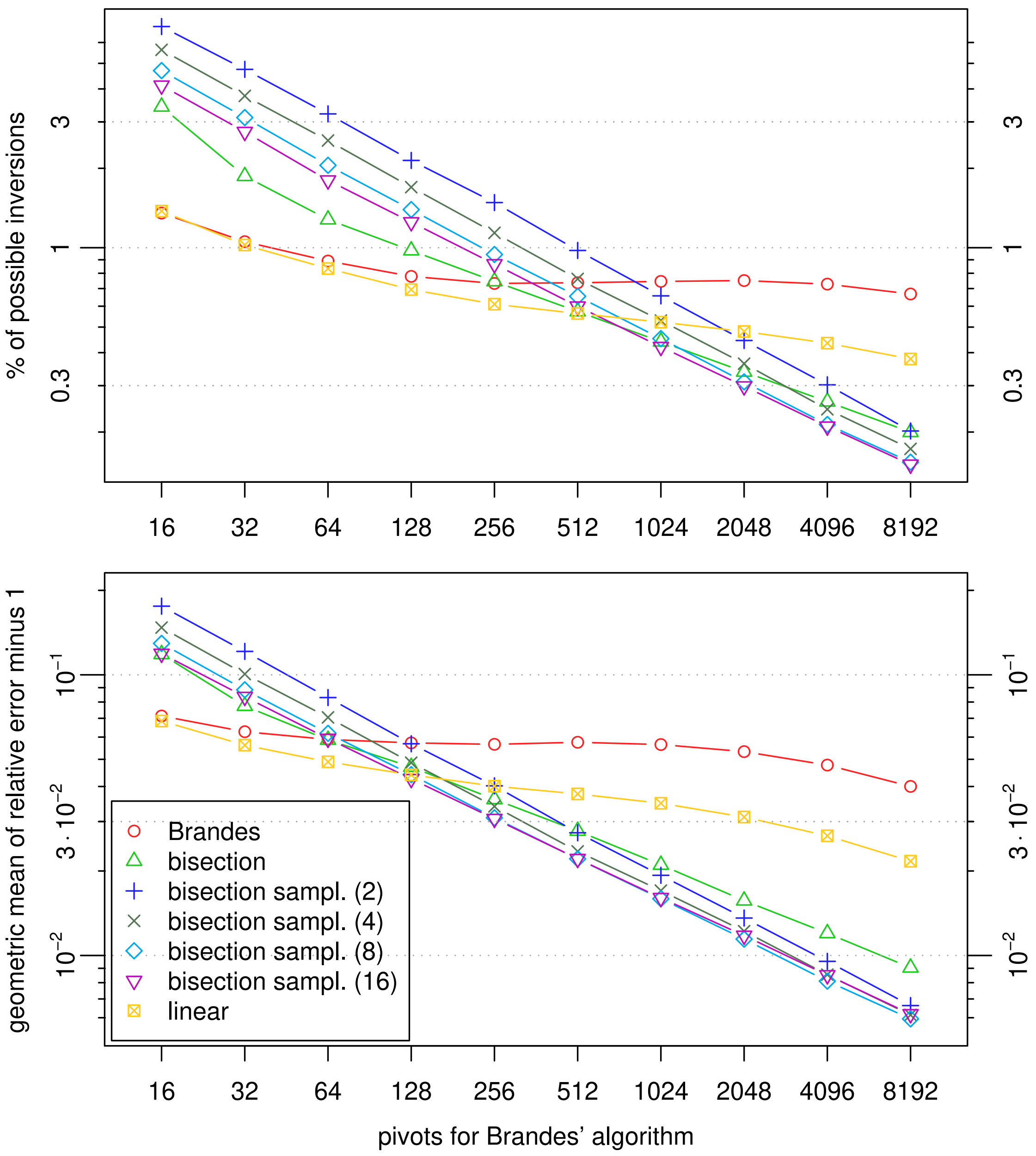


Figure 20: Global scores for general betweenness centrality estimation

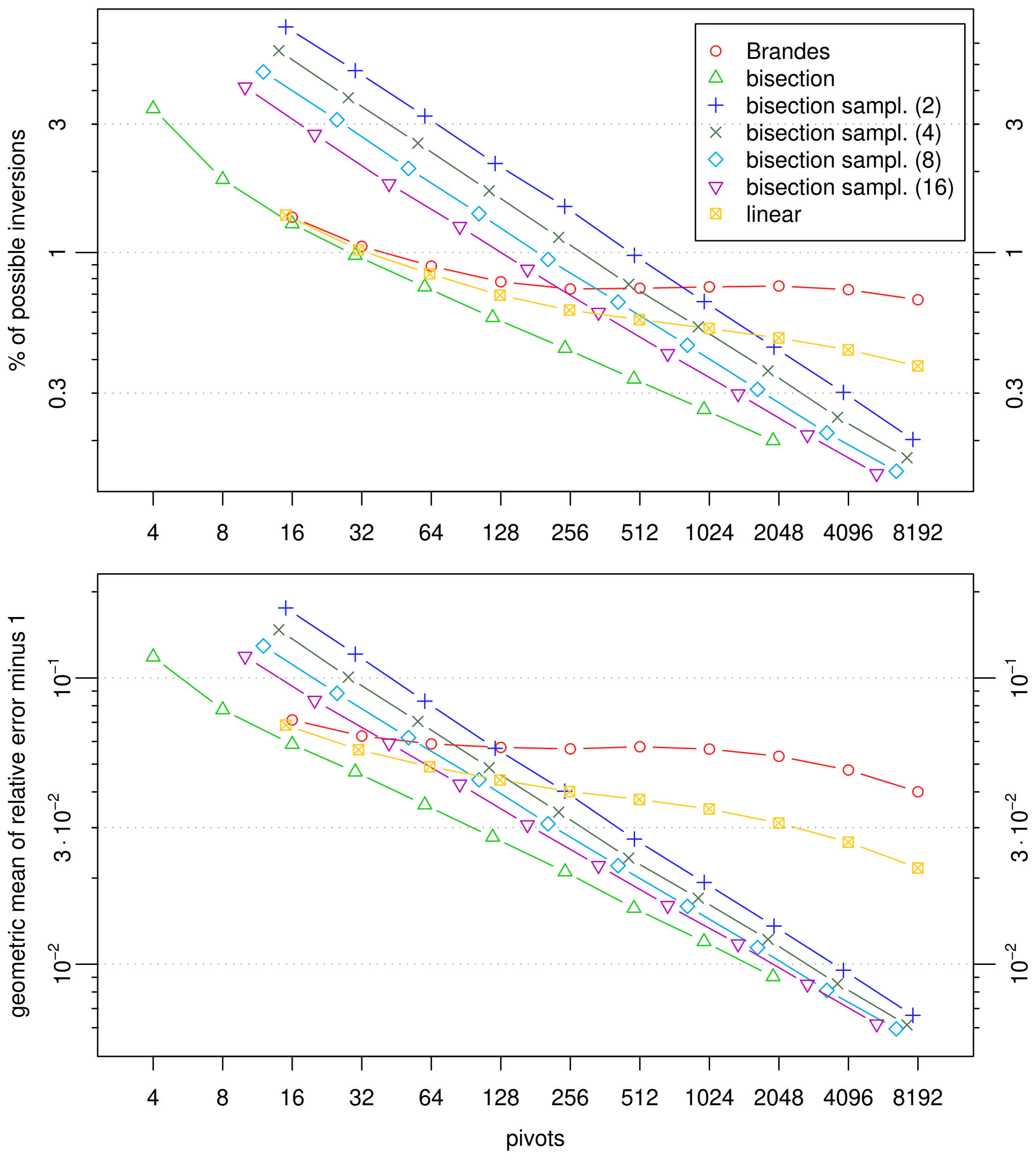
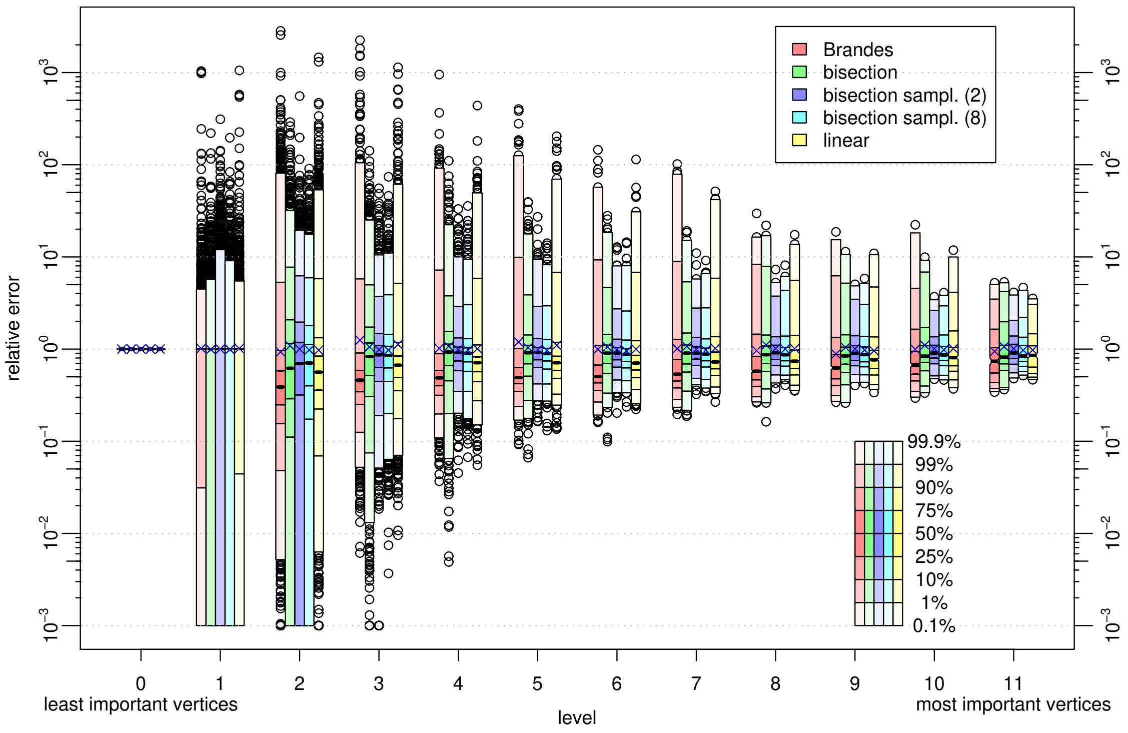


Figure 21: Global scores for general betweenness centrality estimation, real number of pivots as x-value



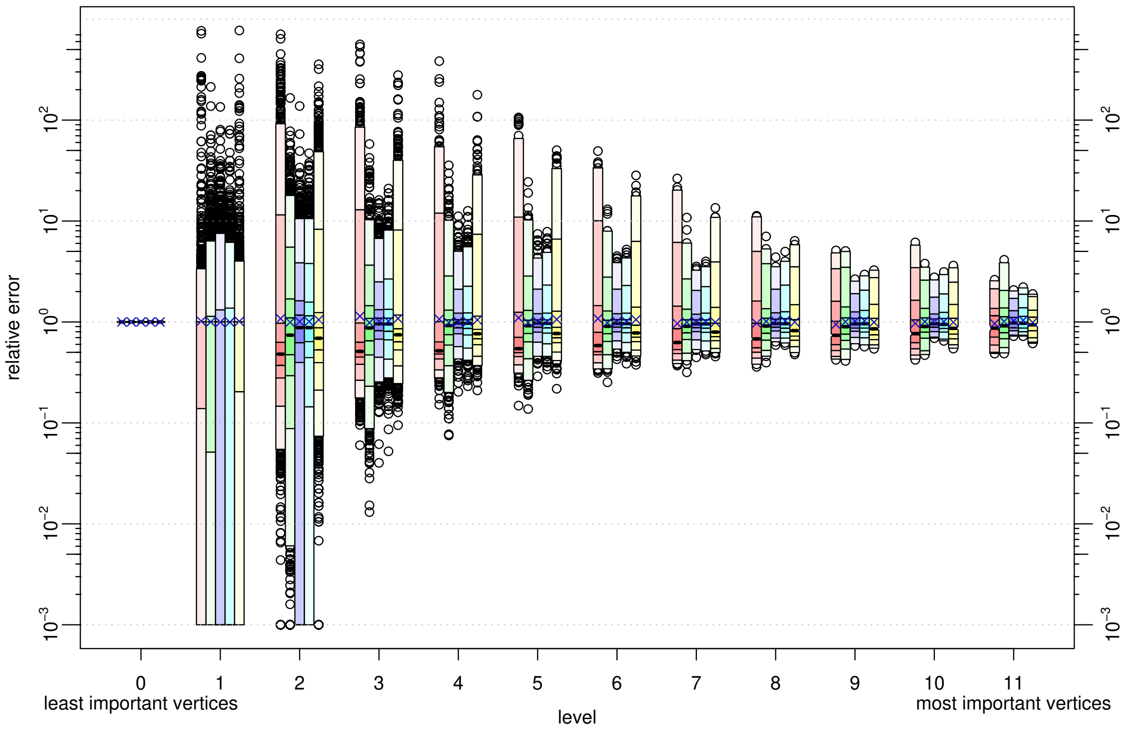
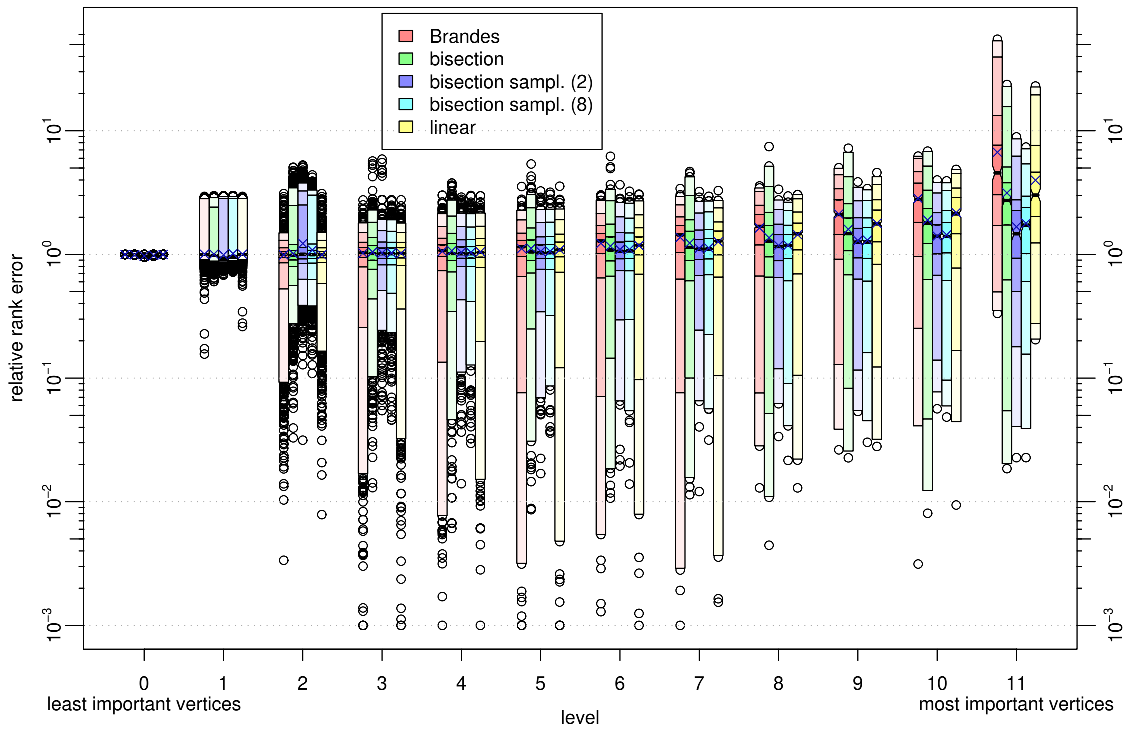


Figure 22: Relative error of general betweenness centrality approximation, classified by level, 32 pivots above, 128 below, boxes represent 0.1%, 1%, 10%, 25%, 50%, 75%, 90%, 99%, 99.9% quantiles, blue cross is mean value, circles are outliers



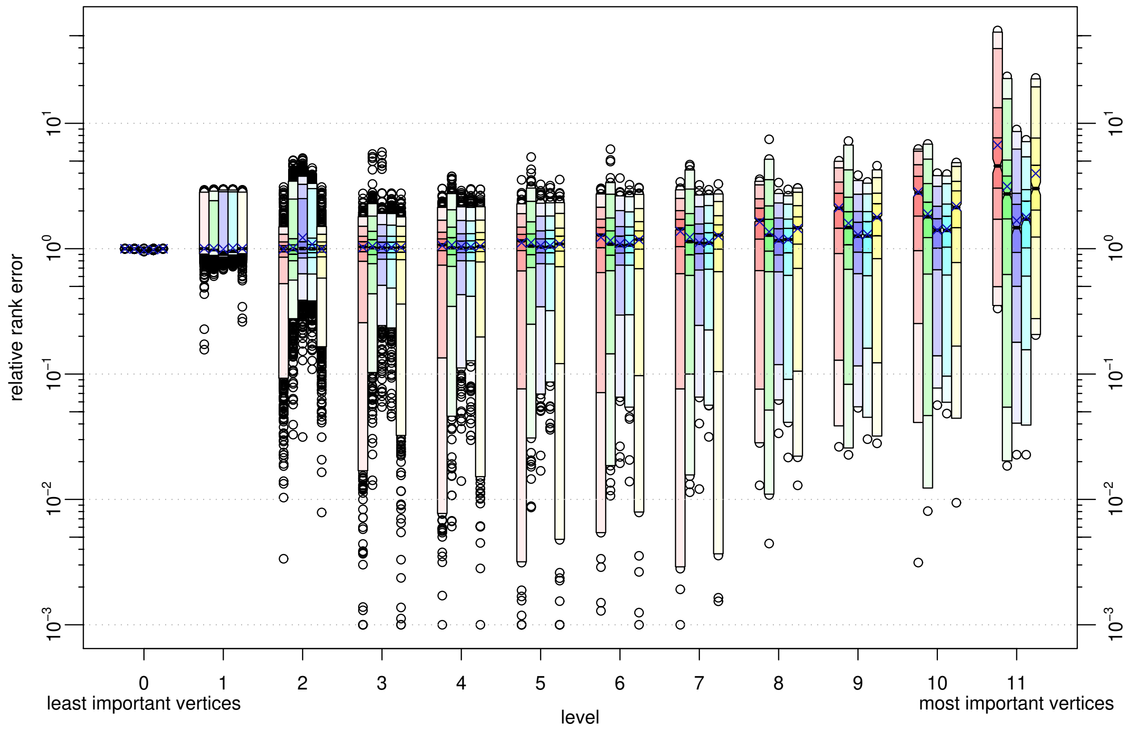


Figure 23: Relative rank error of general betweenness centrality approximation, classified by level, 32 pivots above, 128 pivots below, boxes represent 0.1%, 1%, 10%, 25%, 50%, 75%, 90%, 99%, 99.9% quantiles, blue cross is mean value, circles are outliers

| graph | vertices | edges | source |
|---|---:|---:|---:|
| Belgian road network | 463 514 | 596 119 | PTV AG |
| Belgian road network with unit distance | 463 514 | 596 119 | PTV AG |
| US patent network | 3 774 769 | 16 518 947 | [12] |
| World-Wide-Web graph | 325 729 | 1 497 135 | [14] |
| CNR 2000 Webgraph | 325 557 | 3 216 152 | [8] |
| CiteSeer citation network | 268 495 | 2 313 294 | [6] |
| CiteSeer co-authorship network | 227 320 | 1 628 268 | [6] |
| CiteSeer co-paper network | 434 102 | 32 073 440 | [6] |
| DBLP co-authorship network | 299 067 | 1 955 352 | [7] |
| DBLP co-paper network | 540 486 | 30 491 458 | [7] |

Table 3: Other tested real-world networks

We achieved similar results for other real-world networks, listed in Table 3. The citation, co-authorship and co-paper networks are crawled from CiteSeer of DBLP. A co-authorship network has authors as vertices and edges between them if they wrote a paper together. Opposite is the co-paper network with papers as vertices and edges between them if they share at least one author.

Figure 24 shows the improvement yielded by bisection sampling with 2 samples over Brandes' method. With respect to the Euclidean distance, the bisection sampling method is always better than Brandes' method. But with growing number of pivots the difference gets smaller and smaller. Surprisingly the inversion number shows a completely different picture. An explanation would be that after a few pivots the estimations for the most important vertices are quite accurate, even with Brandes' method, leading to a small distance. However Brandes' estimation of unimportant vertices is still bad. For the inversion number the importance of vertices is irrelevant. The bisection sampling method takes advantage of that and shows increasing difference to Brandes' method. But bisection sampling does not perform that well on all graphs. The directed graphs from Table 3, the US patent network and the web graphs, are more difficult to approximate. Bisection sampling is sometimes better, sometimes worse. Linear scaling is always better, see Figure 25. Shortest path searches turned out to be very local. This means that all approximation algorithms have trouble eliminating false zeros for nodes of small betweenness. This is a disadvantage for bisection sampling since it is slower and since it does not regard all shortest paths. On the other hand, the same effect makes these networks relatively easy for the exact algorithm. For example, our implementation solves the US patent graph in 127 min, only 2.5 times more time than [2] need using 16 IBM-P5 processor cores.

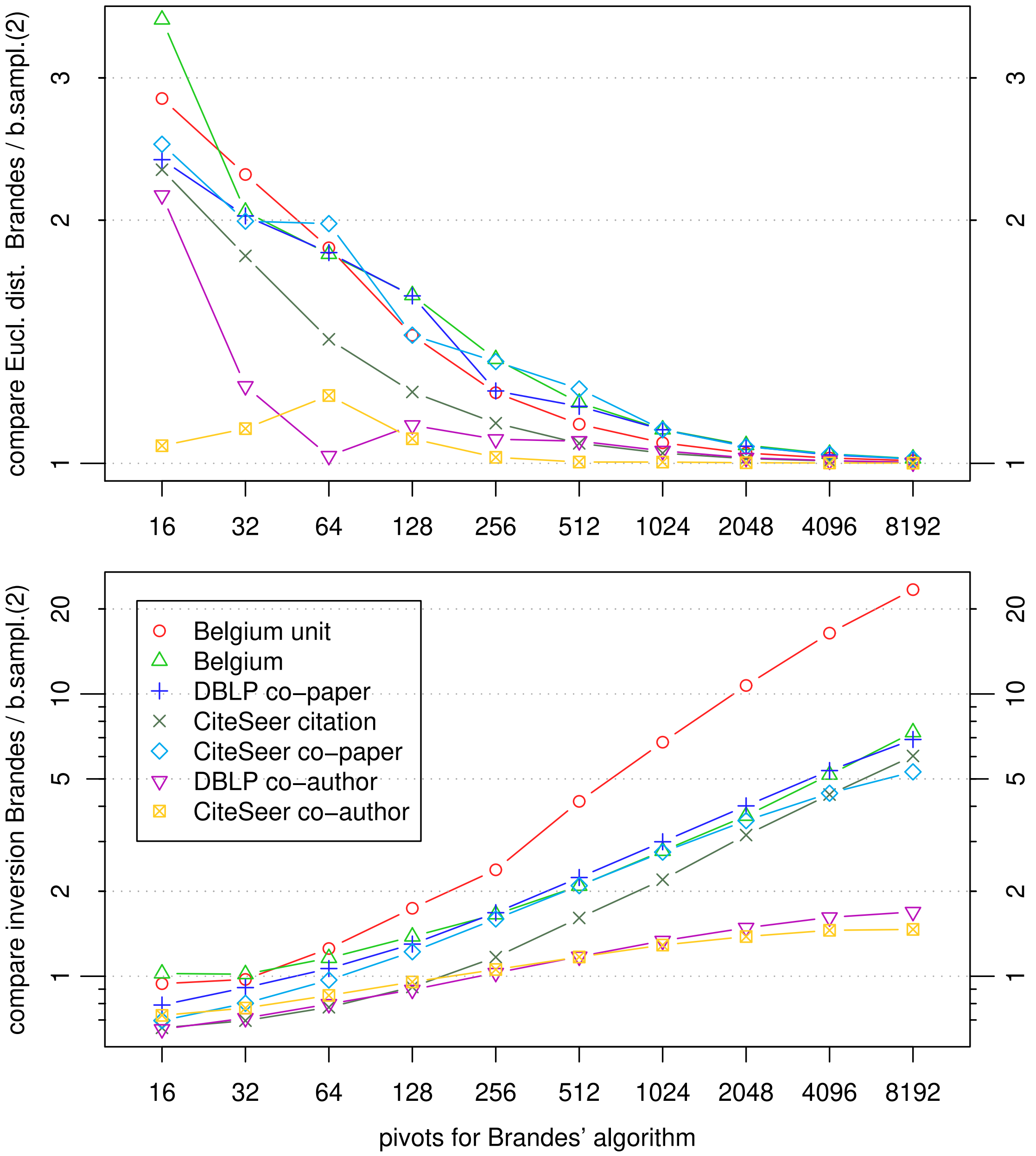


Figure 24: Global scores for general betweenness centrality estimation, Brandes' method compared to bisection sampling with 2 samples

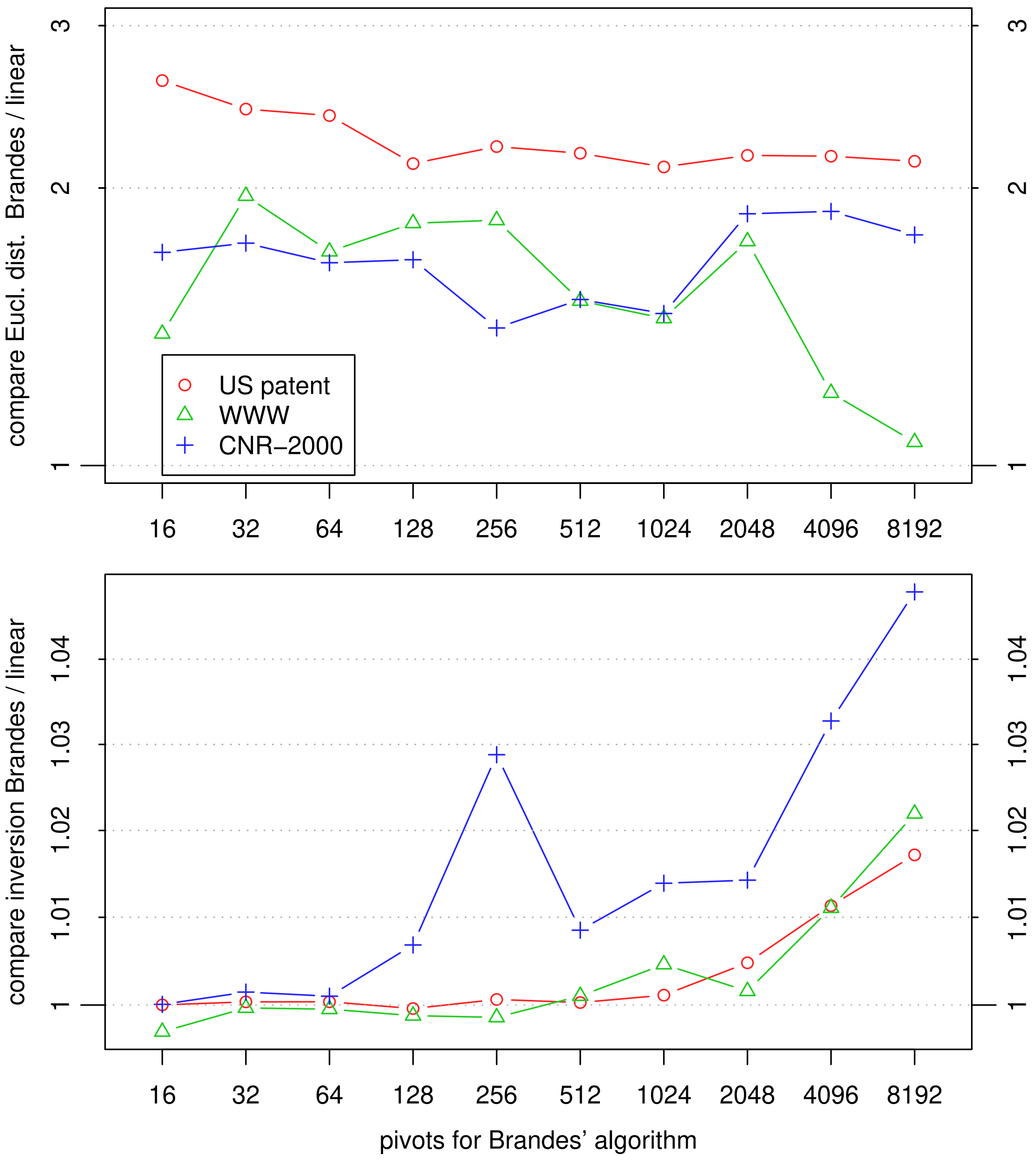


Figure 25: Global scores for general betweenness centrality estimation, Brandes' method compared to linear scaling

## Problems of the Methods

All methods have the same problems as for canonical betweenness centrality.

The bisection method protrudes because of its runtime. Processing the acyclic shortest path graph, the result of the MultipathSSSP, is time-consuming. A tree has not more than $(n-1) = 392\,400$ edges. In Figure 26 you see that for $\approx 25\,000$ pivots the graph is a tree because there are no additional edges. For the remaining pivots there are up to 5 893 513 additional edges, 4 117 080 on average, more than ten times larger than $n$. The runtime of the bisection algorithm comprises of the BFS runtime (the graph has unit distance) and the DFS processing of the acyclic shortest path graph. The BFS runtime is only slightly affected by additional edges whereas additional edges have huge impact to the DFS processing of the graph. With this knowledge, runtimes in Table 2 can be understood.

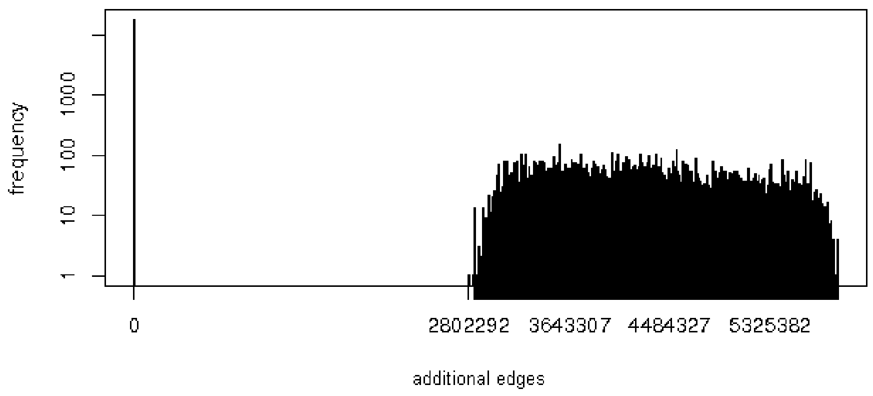


Figure 26: Additional edges in acyclic graph of SSSP. Horizontally number of additional edges, vertically frequency.

## 4 Conclusion

Our new approximation methods *linear scaling* and *bisection* for canonical betweenness centrality and *linear scaling* and *bisection sampling* for general betweenness centrality show better results than Brandes' method. In particular when good approximations of unimportant vertices are required. They are more stable, independent of the number of pivots. Linear scaling is a small modification to Brandes' method with same space requirements. The bisection method for canonical betweenness centrality and the bisection sampling method for general betweenness centrality show best approximation results with time requirements still in $O(SSSP) + O(n)$, but they need additional space.

Robust statistics could be applied to all methods to eliminate outliers. The $q$ smallest and highest contributions are not taken into account. This will probably gain more stable results and limit the maximum error. Local search around vertices with estimated value zero could be used to check whether vertices actually have value zero or not. An approach would be to use all vertices of the graph as pivots but limit the number of settled vertices in Dijkstra's algorithm.

## Acknowledgments

Thanks to Kamesh Madduri and Robert Görke for providing us with test networks. Additional thanks to Kamesh Madduri for providing exact general betweenness centrality values with the networks as well.

# 5 Appendix

## 5.1 Canonical Betweenness Centrality

Figure 27 shows relative error for the Belgian road network. Each dot represents a vertex in the network, x-value is the exact canonical betweenness centrality value, y-value is the relative error.

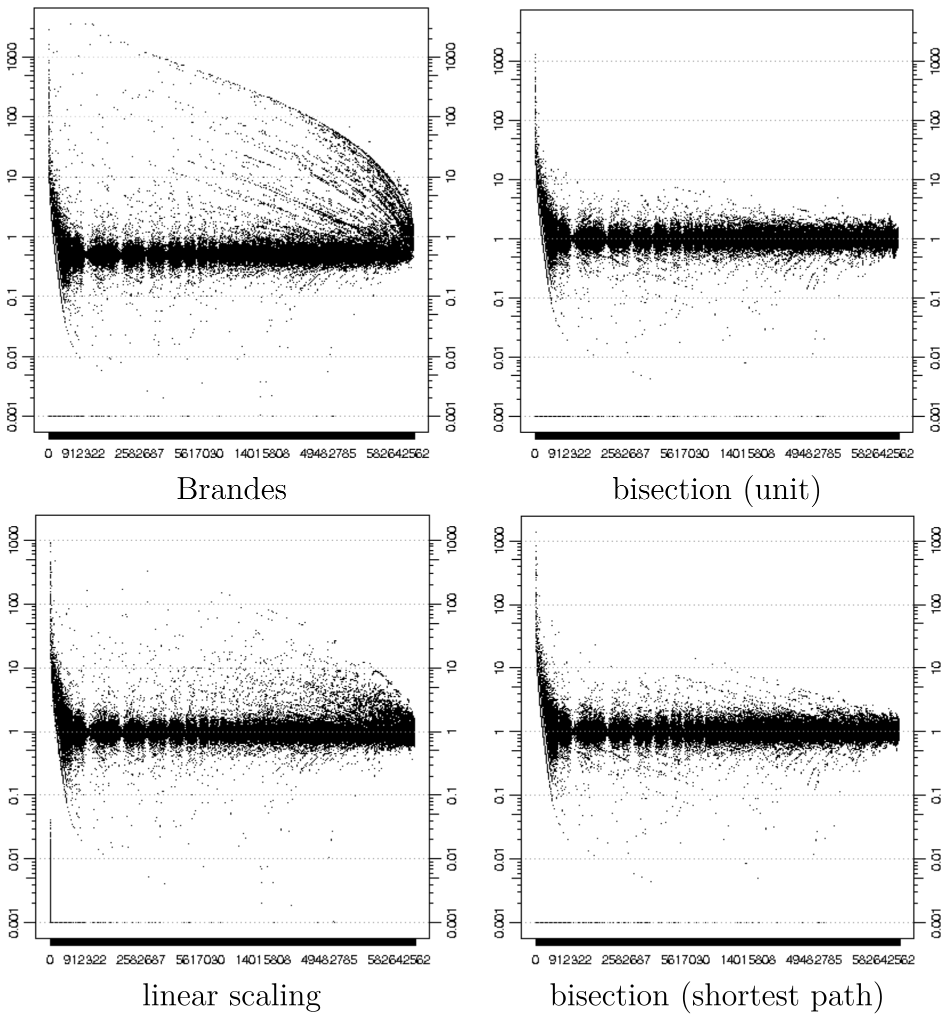
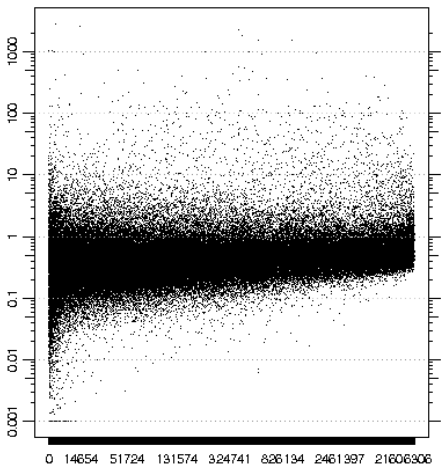


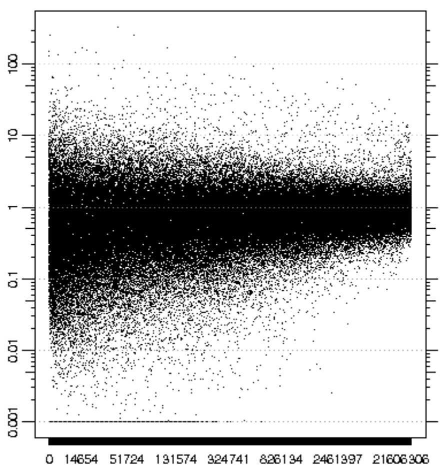
Figure 27: Belgian road network, canonical betweenness centrality, each possible $c_C(v)$ is a distinct level horizontally, relative error vertically, 32 pivots, from left to right, top to bottom: Brandes', bisection (unit), linear scaling, bisection (shortest path) method

## 5.2 General Betweenness Centrality

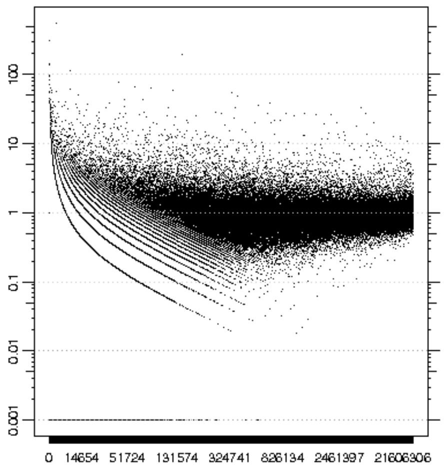
Figure 28 shows relative error for the actor graph. Each dot represents a vertex in the graph, x-value is the exact general betweenness centrality value, y-value is the relative error.



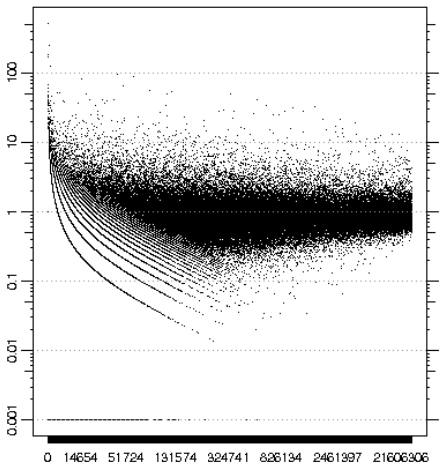
Brandes



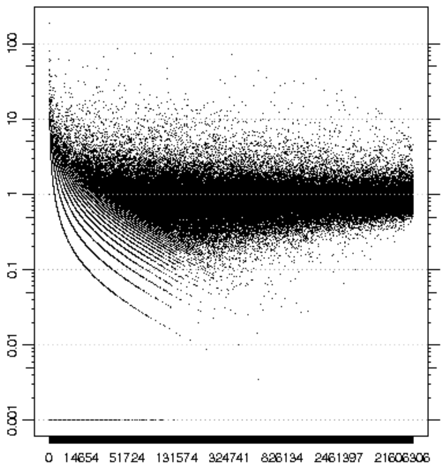
bisection



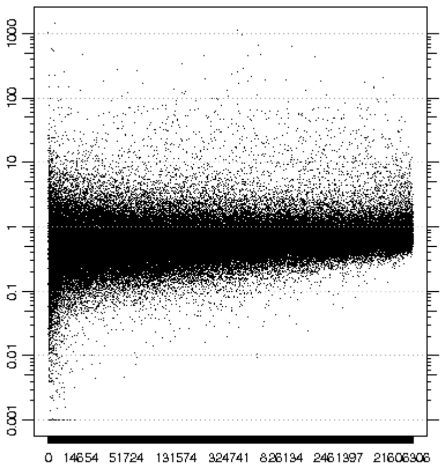
bisection sampling (2 samples)



bisection sampling (4 samples)



bisection sampling (8 samples)



linear scaling

Figure 28: Actor graph, general betweenness centrality, each possible $c_B(v)$ is a distinct level horizontally, relative error vertically, 32 pivots, from left to right, top to bottom: Brandes', bisection, bisection sampling (2,4,8), linear scaling

## References

[1] J. M. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, 1971.

[2] David A. Bader and Kamesh Madduri. Parallel algorithms for evaluating centrality indices in real-world networks. In *ICPP*, pages 539–550. IEEE Computer Society, 2006.

[3] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[4] Ulrik Brandes, editor. *Network analysis*. Springer, 2005.

[5] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, special issue on Complex Networks' Structure and Dynamics, to appear.

[6] CiteSeer. Scientific Literature Digital Library. http://citeseer.ist.psu.edu/, 2007.

[7] DBLP. DataBase systems and Logic Programming. http://dblp.uni-trier.de/, 2007.

[8] Laboratory for Web Algorithmics.
http://law.dsi.unimi.it/index.php?option=com_include&Itemid=65.

[9] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[10] Ronald J. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*, pages 100–111. SIAM, 2004.

[11] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17:57–63, 1995.

[12] A. B. Jaffe Hall, B. H. and M. Tratjenberg. The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools. *NBER Working Paper*, 8498, 2001.

[13] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):713–721, 1963.

[14] Notre Dame CNet resources. http://www.nd.edu/~networks/.

[15] Gert Sabidussi. The centrality index of a graph. *Psychometirka*, 31:581–606, 1966.

[16] Dominik Schultes and Peter Sanders. Dynamic highway-node routing. In *6th Workshop on Experimental Algorithms*, volume 4525 of *LNCS*, pages 66–79. Springer, 2007.

[17] Alfonso Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953.